1999-09

Distributed software applications in JAVA for portable processors operating on a wireless LAN

Rothenhaus, Kurt J.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

DISTRIBUTED SOFTWARE APPLICATIONS IN JAVA FOR PORTABLE PROCESSORS OPERATING ON A WIRELESS LAN

by

Kurt J. Rothenhaus

September 1999

Thesis Advisor: Ted Lewis

Approved for pubic release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Distributed Software Applications in JAVA for Portable Processors Operating on a Wireless LAN	5. FUNDING NUMBERS If Thesis Technical Report. Need funding number here.
6. AUTHOR(S) Kurt J. Rothenhaus	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

As the wave of future Information Technology makes its way into the construction and design of new ships and submarines, it is imperative to examine methods to thoroughly economically backfit older platforms with similar technology. Affordable, Commercial-off-the-shelf (COTS) industrial products have provided us with a means to reduce miscommunication and exponentially increase the availability of information via small pen based computers operating on a wireless LAN. To take full advantage of the communications capabilities of these units and to fill the unique needs of the afloat Navy, the development of software applications is required. These software applications must be effective, tailored, and inexpensive if they are to be made available to older platforms. A JAVA based Intranet is one solution. The simplicity and economy of web-based software coupled with the power and functionality of pen-based computers, creates a dynamic and effectual architecture.

14. SUBJECT TERMS Wireless Local Area Networks, PDAs, Handheld Computers			15. NUMBER OF PAGES 182 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

DISTRIBUTED SOFTWARE APPLICATIONS IN JAVA FOR PORTABLE PROCESSORS OPERATING ON A WIRELESS LAN

Kurt J. Rothenhaus
Lieutenant, United States Navy
B.S., University of South Carolina, 1992

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1999

ABSTRACT

As the wave of future Information technology makes its way into the construction and design of new ships and submarines, it is imperative to examine methods to thoroughly economically backfit older platforms with similar technology. Affordable, Commercial-off-the-shelf (COTS) industrial products have provided us with a means to reduce miscommunication and exponentially increase the availability of information via small pen-based computers operating on a wireless LAN.

To take full advantage of the communications capabilities of these units and to fill the unique needs of the afloat Navy, the development of software applications is required. These software applications must be effective, tailored, and inexpensive if they are to be made available to older platforms. A JAVA-based Intranet is the solution. The simplicity and economy of web based software coupled with the power and functionality of pen-based computers, creates a dynamic and effectual architecture.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author would like to acknowledge the financial support of Gary Lacomb at the New Attack Submarine Program, PMS450T2 and Tom Hazzard of Navsea IDEA in sponsoring this thesis effort.

The author would also like to thank Professors Ted Lewis and Xiaoping Yun for their guidance and support during this research effort.

Finally, the author would like to thank his spouse, Momoko, and his children, Mia and Nina, for their patience and understanding during the preparation and conduct of this research.

I. INTRODUCTION

Communications and computing systems on US Navy ships and submarines continue to be stove piped in nature and hardware specific. They prevent sharing of data and dramatically increase personnel workload. Unfortunately, the information technology gap between ship and submarine crews and their interfaces to the shore and industrial establishment is increasing. This gap is caused in large part by a lack of open system software available to US Navy afloat forces. Some of the central causes are the draconian Department of Defense software procurement/development regulations, and the high costs associated with procurement of DoD compliant software.

In an effort to reduce costs and increase availability of software, program offices have embraced commercial-off-the-shelf (COTS) products. But, the unique requirements of afloat software applications preclude a "one-to-one" mapping of commercial application to DoD requirements. In addition many of the tasks that would benefit mostly from automation are in locations and environments not suited for traditional processors. To automate these mobile tasks a computer should possess the following qualities: continuous connectivity with the network, be portable and comfortable, and utilize the same operating system as the rest of the computing environment.

The problems facing afloat information systems are mobility and cost, which can be solved by careful selection of software running on the newly emerging wireless local area network (LAN) architectures. Mobility is ensured by the wireless LAN, and the low-cost requirement is ensured by leveraging commodity Internet technology in the software.

Therefore, it is the thesis of this work that a Java-based distributed Intranet architecture can support low cost, mobile afloat computing requirements by utilizing current development tools. Because these are COTS products, we can improve communication and exponentially increase the availability of information via small pen-based portable and wearable computers operating on a wireless local area network (LAN) at a minimal cost. Therefore the simplicity and economy of web-based software, coupled with the strength and functionality of pen-based computing, creates such a dynamic, effective architecture, it seems inevitable that this combination should be exploited for most computerized, non-tactical, afloat administrative activities.

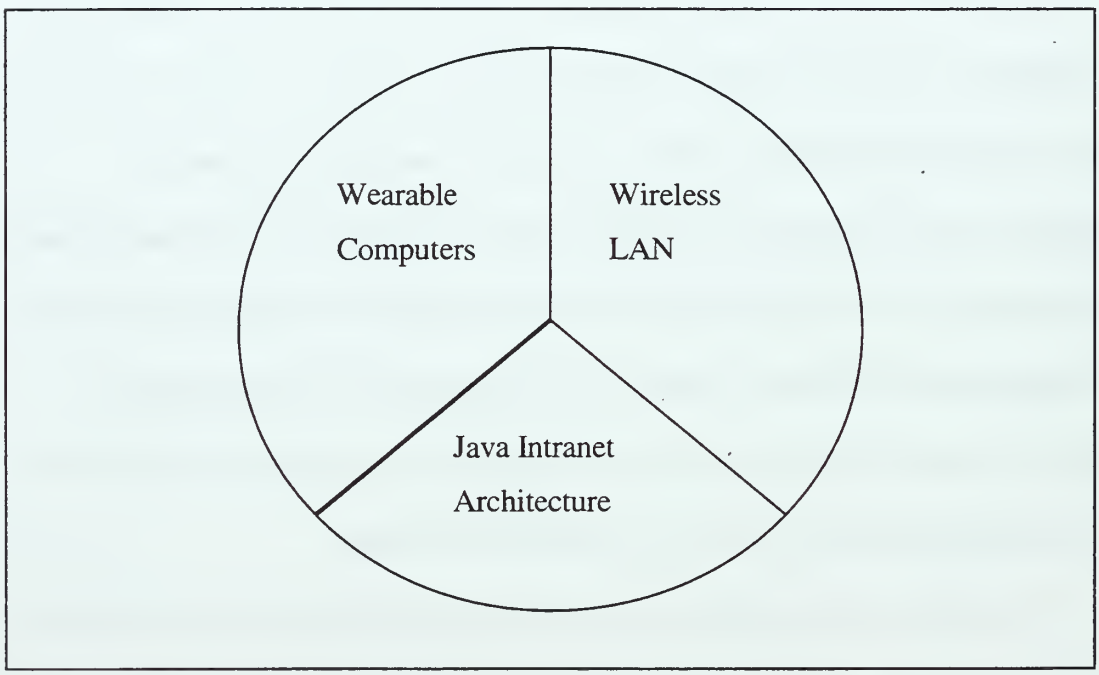

Figure 1. The Three Central Technologies.

The problem addressed by this thesis is to find suitable software architecture for developing applications onboard ships and submarines. This can be done by combining Internet-based software with wireless LAN hardware and hand held or wearable computers.

A. GOALS FOR THIS THESIS

The central goal of this thesis is to demonstrate, by means of a working prototype and other metrics, that a Java-based Intranet operating on a wireless LAN with wearable and/or pen-based clients is an effective method to automate numerous personnel functions that have previously evaded automation. The prototype software will be completely "client side," downloadable, require no prior configuration, and exclusively utilize commercial-off-the-self (COTS) products. Prototype applications will include the following modules: (1) damage control communications, (2) maintenance management and interactive technical manuals, and (3) preventative maintenance instructions. The thesis will examine the revision and extensibility features of Java, as well as the benefits of utilizing standard query language (SQL) databases as the primary "data structure" in the applications. Finally, ship and submarine operational tests will be conducted and analyzed by the author for issues in usability and functionality.

B. WIRELESS EQUIPMENT IN THE MEDICAL INDUSTRY

The medical industry has led the commercial sector in the integration of wireless LAN and portable processor technology. Driving the trend towards wireless in health care

is the need for accurate and timely information in a mobile environment. Another significant factor is the increase in support staffs required by complex insurance billing and the general litigious nature of the medical industry demanding precise documentation. The hospitals found they were able to increase efficiency by having the caregivers enter patient information, order tests and prescribe treatment, with everything documented instantaneously. The two main challenges to integrating wireless has been the legacy software systems of the hospital and the previously slow voice recognition software available. Many future developments in wireless will continue to be driven by the medical industry, as applications such as Tele-medicine and improved streaming video and biometrics allow fewer people to provide more care[1][8].

C. THESIS OUTLINE

The remainder of the thesis is as follows. Chapter II discusses the various tasks that lend themselves to this particular architecture, and outlines a specification for the prototype software. Chapter III details the benefits of the proposed system architecture, to include the attributes of Java's Database capabilities, to create distributed systems that are ideal for afloat users. Chapter IV examines the various processor platforms available on the market as well as the wireless LAN components available for connectivity. Chapter V details the prototype software architecture and interface. Chapter VI discusses the results and methods of shipboard testing conducted by Professor Xiaoping Yun and the author. Finally, Chapter VII discusses conclusions and recommendations for further research. Appendix A and B contain the Java source code for all applications.

II. SOFTWARE SPECIFICATIONS

Currently, the US Navy utilizes an antiquated but reliable method of inter-ship communications for its older afloat platforms. Stovepipe in architecture, inter-ship communications of both mundane status and critical command and control are passed utilizing the same method. During a casualty or emergency, crewmembers form into various teams and fill positions to place the vessel in a higher state of readiness. A good deal of these positions are dedicated exclusively to relaying communications to and from various control centers. Verbal communications are accomplished via sound powered phones or hand-held radios. These methods, although reliable in most cases, flood decision-makers with information, and may obscure recognition of critical data. Currently, most platforms maintain a "status board" of some type. (This board is usually updated manually by a crewmember dedicated to nothing else!) The accuracy and timeliness of the status board is dependent upon the knowledge and ability of the individual to pick relevant information off the cluttered voice circuit.

The stovepipe nature of shipboard inter-communications is also illustrated by the way repair manuals, preventative maintenance instructions, supply inventory, and equipment parameter logs are processed. Each system maintains its data in a different, and often propriety format, unable to share information with the other systems. With no common database, the systems require redundant data entry and maintenance. Lack of consistency between the numerous systems obligates time consuming and inaccurate

reconciliation. The inability to share data and updates with other systems, and the storage of data in propriety formats prevent older platforms from entering the digital age.

The afloat environment is littered with countless opportunities to leverage current practices with information technology to improve timeliness and accuracy. A great deal of these tasks eluded automation by their very nature. However, as software and hardware technology matured more and more options for automation became available. Currently wearable and pen based processors can effectively operate on a wireless LAN providing computing services on a common operating system such as Windows 95/NT or Linux. These devices can run virtually all applications currently in use by the Navy to include ATIS (tech drawing viewer), PMS automation programs, SNAP II and other applications designed to run on Wintel processors. By utilizing wearable or pen-based processors that support Windows 95/NT we ensure compatibility with existing software, as well as new products and concepts brought forward by the commercial market.

Due to the specific nature of afloat US Navy non-tactical applications we cannot rely on the commercial sector to provide automation for many tasks. Obviously, there are tasks that the Navy shares with the civilian sector that the commercial market provides well for, such as word processing and spreadsheets. However, many of the tasks Navy personnel perform are extremely localized, even to the platform class level, hence eliminating the probability the commercial market will provide a solution. In the past the Navy had to choose between either buying expensive software generated through the costly Department of Defense (DoD) acquisition cycle or perform the task without automation. To automate these Navy specific tasks software that can run on

various hardware platforms utilizing existing assets is required. The software also needs to be relatively inexpensive to create and maintain as well as capitalize on computing skills the user has attained in every day life by using familiar and common interfaces. The trade-of in cost and ease of use is usually found in robustness. This loss in robustness would be unacceptable in tactical and control systems, but is acceptable in the type of administrative tasks this thesis proposes to automate. An appropriate level of robustness and reliability for the software would be equivalent to that found in mid-size corporations operating software "important but not life critical."

In the remainder of this chapter the specification for the prototype software will be detailed. The subsections are broken down by software module. The software specifications provided are tailored to providing a prototype that provides a proof of concept and not the detailed specifications a final application would require. The modules would operate under a common operating environment and maintain a consistent human computer interface.

A. DAMAGE CONTROL APPLICATIONS

The ship and submarine damage control environment is the most demanding of the various software modules. Used under stress in a critical manner, the application pushes the envelope of robustness and reliability to the acceptable limit afforded by the planned intranet/SQL style architecture proposed. The system must use a simple graphical interface and update fast enough to support the critical nature of damage control information. It is

essential that the application provides spatial representation for the location of the casualty and relay pertinent data from various sources in a concise and readable manner.

The application will receive its updates from a series of wearable and pen based processors connected via a wireless LAN. The application must therefore be distributed in nature. The goals of the software module are detailed as follows:

Task Analysis:

Primary task 1: Provide a 3D image of casualty location.

- **Subtask 1.A:** Allow user to zoom in or out.
- **Subtask 1.B:** Display surrounding casualty assets.
- **Subtask 1.C:** Display fire/flooding/NBC boundaries.
- **Subtask 1.D:** Display material readiness of zone.

Primary task 2: Display status of casualty response.

- **Subtask 2.A:** Display readiness of response team.
- **Subtask 2.B:** Display members of response team.
- **Subtask 2.C:** Display status of casualty.
- **Subtask 2.D:** Display material readiness.
- **Subtask 2.E:** Display OBA timer status.
- **Subtask 2.F:** Display Halon activation time.
- **Subtask 2.G:** Display bilge sprinkler time.

Primary task 3: Allow configuration management.

- **Subtask 3.A:** Enter fire team members name.
- **Subtask 3.B:** Enter hospital corpsman name.
- **Subtask 3.C:** Enter OBA time limit
- **Subtask 3.D:** Team leader

B. MAINTENANCE MANAGEMENT APPLICATIONS

The US Navy utilizes a large database of maintenance requirements stored in a system known as the Maintenance Resource Management System. This system when it was first introduced to the fleet in the mid 1970's was cutting edge in both functionality and interface. Today, however even with the introduction of an IBM PC compatible version the software is rigid and difficult to use. The system is divided into three major components based on user group. The three components are a *maintenance broker, repair facility,* and *shipboard.* Each entity maintains its own separate database that is updated from the originating shipboard database. Below is a diagram of the current method shipboard repair requests are process.

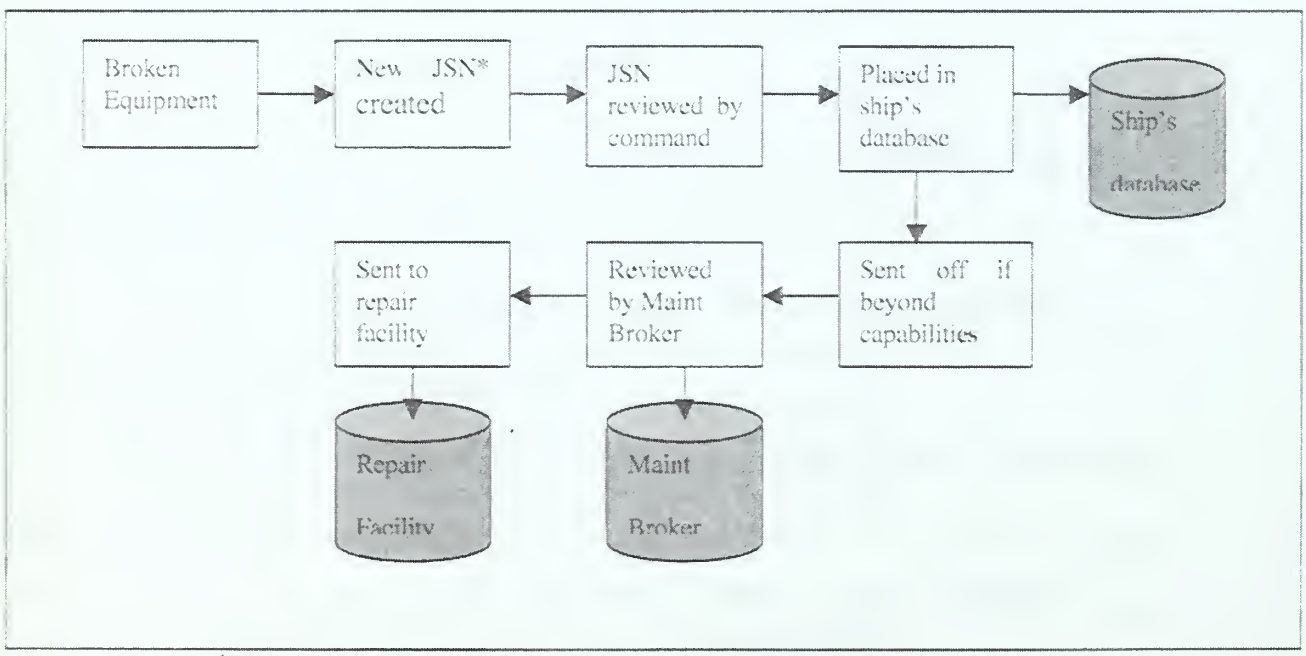

Figure 2. Current Maintenance Action Flow Chart

With current levels of connectivity it would appear to be more cost effective to maintain a common database that all components work from. The prototype system proposed would provide a graphical interface to allow the user to update and create new repair requests. The application could either interface with the current database or a new one. The application must continue to work seamlessly with the other components, as well as maintain a similar arrangement as the old system to reduce user training. The goals of the software module are detailed as follows:

Task Analysis:

Primary task 1: Create New JSN's
- **Subtask 1.A:** Open JSN form.
- **Subtask 1.B:** Automatically assigns JSN.
- **Subtask 1.C:** Utilize expert system to assist in write up.

Primary task 2: Update JSN.
- **Subtask 2.A:** Allow modification of existing JSN.

Primary task 3: Sort Database.
- **Subtask 3.A:** Sort by JSN number.
- **Subtask 3.B:** Sort by work center.
- **Subtask 3.C:** Sort by keyword in blk 35 of JSN.
- **Subtask 3.D:** Sort by command UIC.

Primary task 4: Export Database.
- **Subtask 4.A:** Allow site administrator to update central shore database.

C. WATCHSTANDER AND EQUIPMENT LOGS

Optimally, the concept of a crewman manually recording equipment parameters will someday soon be unnecessary. The ability of systems to perform self-diagnostics and alert a central control system of its impending failure should allow the crew of the future to spend its time in more productive pursuits. In fact, most new platforms are incorporating these new technologies with great success. The problem still exists however in older platforms. Most ships and submarines currently in commissioned service continue to have a dedicated crewman that walks through spaces and records equipment parameters on a paper form. The information the watchstander gathers allows managers the ability to ascertain if equipment parameters are out of specifications. Due to the nature of the paper medium, however, it fails to provide dynamic trend analysis. Hence if a ship or submarine is going to continue to commit a watchstander to gather data, a digital medium should be utilized, with applications and capabilities to use the data. Currently the Navy is not seriously evaluating the option of upgrading older platforms with remote sensing capabilities: hence any other option should be inexpensive enough to compel decision makers to action.

In order to reduce training and add legitimacy, the system interface should be much like the process utilized today. A pen-based processor can display the log sheet in a manner consistent with the logs current format and allow hand writing inputs much like the current system. The data can then be used in trend analysis, failure analysis and other tasks. The goals of the software module are detailed as follows:

Task Analysis:

Primary task 1: Start a new log

Subtask 1.A: Record equipment parameters.

Subtask 1.B: Correct mistaken entries.

Subtask 1.C: Make comments on out of parameter readings.

Primary task 2: Provide digital signature.

Subtask 2.A: Use digital signature for Supervisor.

D. SUPPLY INVENTORY APPLICATIONS

No shipboard community has embraced information technology more than the supply department. Tracking thousands of parts, and handling hundreds of new requests a day, the supply community articulated its requirements and retained a software system ahead of its time. Unfortunately all of this happened in the late 1970's and early 80's. Today, the software and hardware systems utilized are aging obsolete when compared with the systems utilized by the commercial sector. The supply systems command has made some improvements to the current state, but have done little more than adapt the software to a WINTEL vice Digital (DEC) hardware platform.

The dividing lines between the maintenance management, supply inventory, and techmanual applications are fuzzy at best, because of the complementary nature of the tasks. Any future system should be integrated to follow the natural path from maintenance

requirement discovery to repair part purchase. The system must include the management and administrative tasks, but should be focused on the lowest level user, who orders a part or consumable item. The system should also provide relevant statistic's and generate required reports. The goals of the software module are detailed as follows:

Task Analysis:

Primary task 1: Purchase part

Subtask 1.A: Display location and purchase part.

Subtask 1.B: Provide for supervisor approval.

Primary task 2: Provide statistics.

Subtask 2.A: Display current budget.

Subtask 2.B: Display projected expenditures.

Subtask 2.C: Provide breakdown by department.

Primary task 3: Provide automated reports.

Subtask 3.A: Provide automated comptrollers report.

Subtask 3.B: Provide automated CO's report.

Subtask 3.C: Provide automated department reports.

E. REPAIR MANUAL AND EXPERT SYSTEM APPLICATIONS

The proliferation of online support for hardware and software systems has been demonstrated to significantly reduce expenditures in customer support. The repair

manual and expert system applications are a classic example of a one to one mapping of COTS solutions to defense requirements. The drive to reduce industry customer support costs has provided a number of open system solutions that the DOD can adopt. In fact, the widespread use of adobe acrobat software to promulgate written instructions has been tremendously successful at providing current documentation and has reduced the administrative burden of maintaining these documents significantly.

The next step is to provide maintenance documentation online as well. Utilizing a hypertext format parts procurement as well as more detailed information can be provided as required. This allows for a seamless transition from problem discovery to resolution.

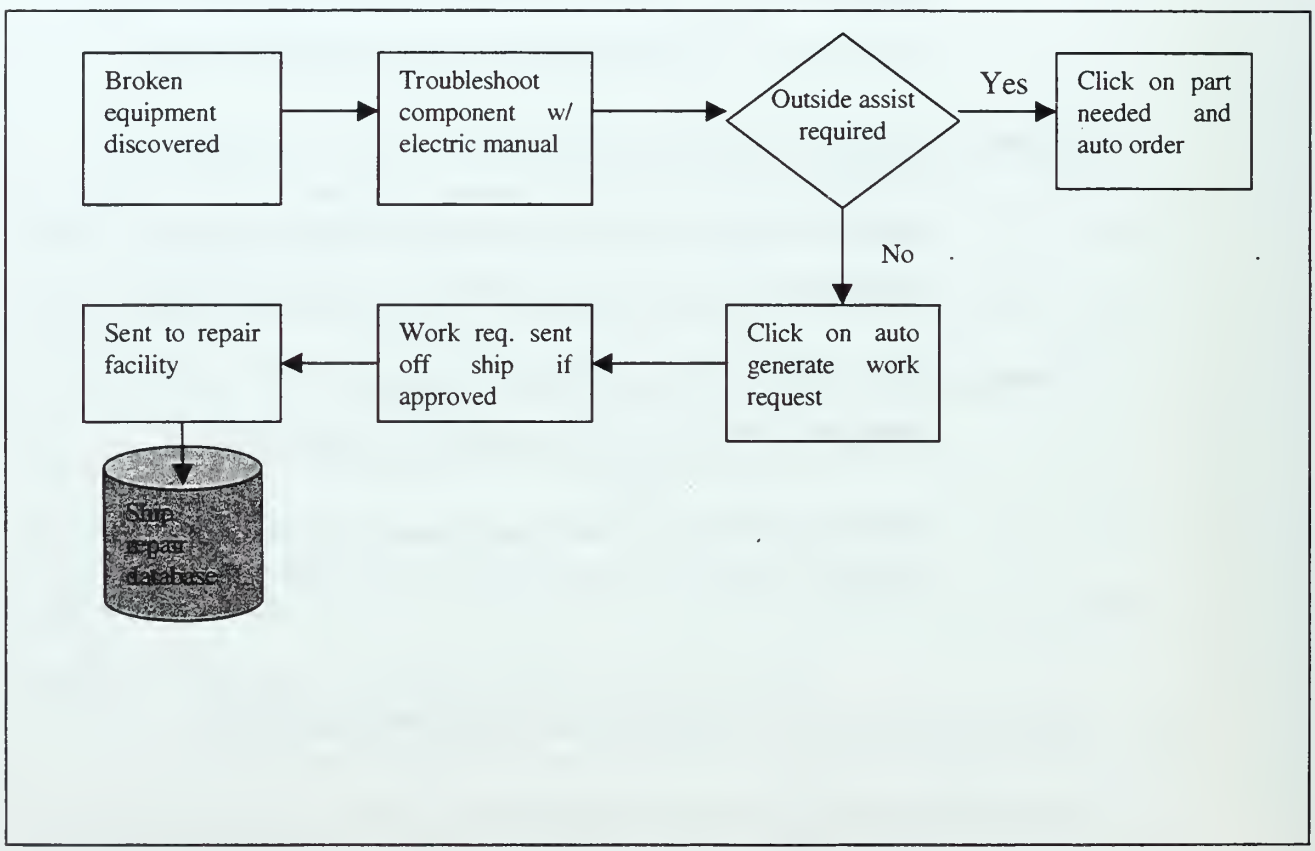

Figure 3. Seamless Maintenance Resolution Matrix

Task Analysis:

Primary task 1: Display Repair Manual

Subtask 1.A: Allow user to zoom in and out.

Subtask 1.B: Display parts information.

Primary task 2: Link to other tasks.

Subtask 2.A: Allow user to click a part and order it.

Subtask 2.B: Auto-generate off ship work request when directed.

F. PREVENTATIVE MAINTENANCE APPLICATIONS

The Navy manages a great deal of equipment from small valves to million dollar gas turbines. All these devices share a common maintenance scheme, the Preventative Maintenance System. An administratively intense method that tracks maintenance accomplishment, and provides detailed written procedures for every action, the Preventative Maintenance System (PMS), is the cornerstone of the Navy's material readiness.

The PMS operates much in the same way it did for the last three decades. The ship either receives or prints thousands of cards with the procedures on them, along with the required supporting documentation. A supervisor schedules the maintenance with respect to the ships schedule and tracks what is accomplished and what must be postponed. A number of statistics on the system is tracked to allow management a yardstick for material readiness.

The Preventive Maintenance System is an ideal candidate for automation. Opportunities for significant labor and material savings as well as improved accuracy are in abundance. Eliminating the time consuming process of keeping the cards up to date, printing new cards and tracking maintenance accomplishments manually can significantly free up wasted time. The goals of the software module are detailed as follows:

Task Analysis:

Primary task 1: Display maintenance procedure

Subtask 1.A: Display desired procedure.

Subtask 1.B: Link to supporting documentation.

Primary task 2: Provide statistics.

Subtask 2.A: Display current accomplishment rate.

Subtask 2.B: Display archived accomplishment rate.

Primary task 3: Track accomplishment and provide automated scheduling.

Subtask 3.A: Indicate status of check (accomplished/In-progress/scheduled/ deferred)

Subtask 3.B: Auto schedule future checks.

Subtask 3.C: Track time, date and crewman who accomplished the check.

F. CHAPTER SUMMARY

The next generation Navy Destroyer (DD-21) has a crew requirement of 300, if the current manning model is used. However, Navy leadership is attempting to reduce the manning to 95. Leveraging information technology is one of the critical steps to reducing the crew requirement, but the old way of developing and acquiring software will not work. Software that requires an army of IT personnel to constantly tend to it will create as much work as it saves. One such solution is to develop software in the model specified in this chapter, and utilize as many standard modules as possible.

III. POTENTIAL FOR A JAVA-BASED INTRANET SOLUTION

In 1990 Sun Microsystems began project "green," a software platform designed to support embedded microprocessors. Green, based on the high order language C++, was soon supplanted by "Oak," a new language that eliminated some of the error prone aspects of C++, such as operating overloading and pointer arithmetic. From Oak came Java, a language portable enough to run on a number of different processors and tailored to the creation of graphical user interfaces (GUI) [3]. The power of Java as a cross platform language was immediately demonstrated by its popularity in supporting applications running on the Internet. Currently Java, and Java Script, a variant of Java designed to be embedded in HTML code, supports the majority of electronic or E-commerce, as the commercial sector turns to the Internet for business.

Since early 1990, the US Navy has continually sought to leverage information technology in an effort to reduce personnel requirements for afloat forces. These efforts have been hindered by the mobile and hostile nature of the afloat environment. Today, wearable computers operating on a wireless LAN have surmounted the hardware component to the technological hurdles to truly integrate user and system, with the only remaining obstacle being Navy tailored software applications. The Java architecture provides a compelling argument to span this software hurdle.

Strongly supported by the commercial sector with an ever-increasing market share, Java combines the power of C++, the structure of Ada, and portability unmatched by any other commercial language. Utilizing the capable standard query language (SQL)

protocols supported by JAVA, a tailored JAVA "Applet" can interface with many propriety database, and allow the Navy to continue to utilize current data storage methods. Ultimately, a "master database of the boat" can be created with hundreds of various Java applications sharing and updating the common database, but until that concept comes to fruition, Java can create a virtual common database.

To take full advantage of the communications capabilities of these units, and to fulfill the unique needs of the afloat Navy, the development of software applications is required. These software applications should be effective, tailored and inexpensive if they are to be made available to older platforms. It is important to note early, that although Java and portable processors are extremely robust, they currently lack the stability and security of more custom tailored solutions, and hence, at the time of this writing, are only appropriate for non-tactical applications. *Although the architecture is ideally suited for administrative and support activities, it is not currently suited for control or weapon systems*. Nevertheless, the Navy has a great deal of tasks that would be greatly enhanced and streamlined with a Java Intranet solution.

A. JAVA ARCHITECTURE

Java can support two types of software architectures; distributed and standalone. In the standalone architecture, Java syntax or code is compiled and turned into *Java byte code*. This code is then interpreted by the *Java Virtual Machine*, a processor specific translator that takes the native byte code and turns it into commands appropriate for the processor. The second architecture Java supports is a distributed or *Applet* model where

the byte code is transmitted to a client processor via a network, where the transmitted byte code is then run by the client's virtual machine. Ideally suited for distributed applications the Applet code can run on any platform supported with a Java virtual machine [3].

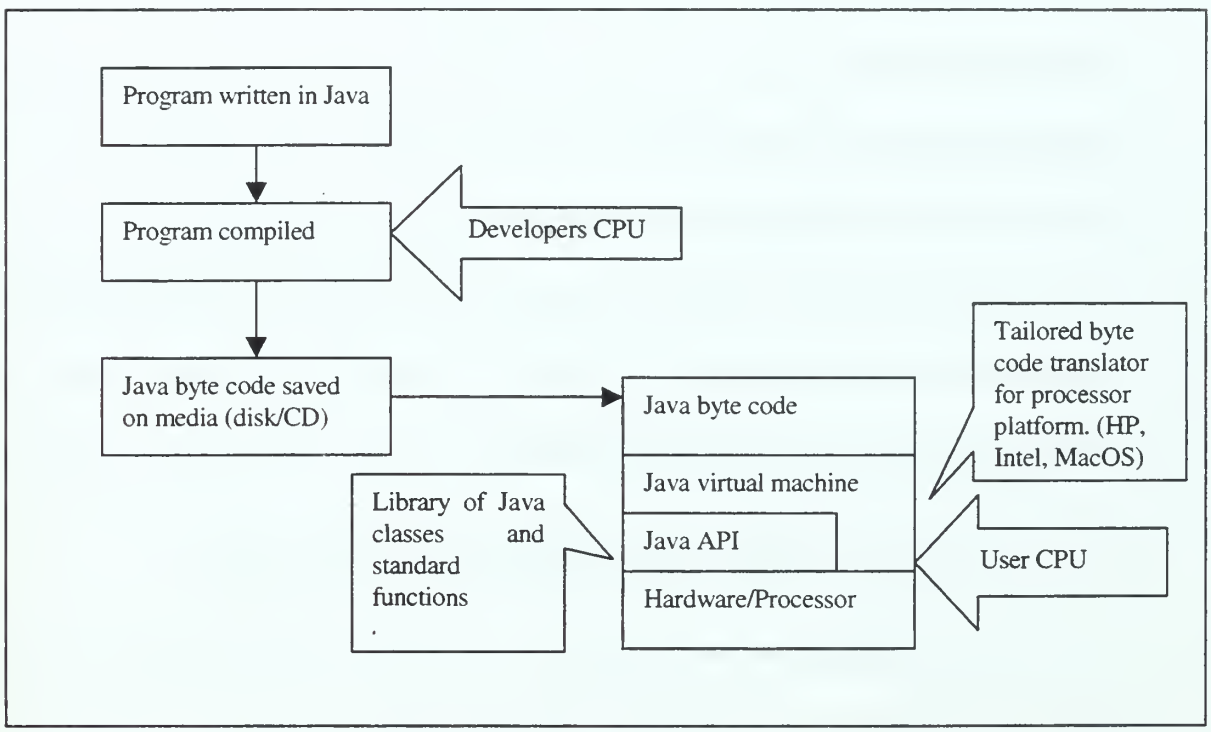

Figure 4. Java Application Architecture

Java is a full function, object oriented language that supports multi-threaded robust applications. Sun Microsystems has eliminated many of the pitfalls that caused many of the

quality problems in the C and C++ languages such as operator overloading, multiple inheritance and manual memory management. In the past, automatic memory management or "garbage collection" was viewed as dangerous to real time systems, as the program could execute a "garbage collection" routine during a critical process. However the speed of today's processors makes this an unlikely problem, especially for the prototype software in this thesis.

The Java language classes or API, provides a library of standard functions and routines compiled and optimized by Sun. These API are well documented and fiercely maintained by Sun Microsystems. Sun also avoids the slow response times for updates to the language and clarity problems that are normally associated with open source or committee controlled language specifications, by diligently maintaining the integrity of Java. Sun has endeavored to keep the size and complexity of the virtual machine small enough to run on small processors, which makes the Java ideally suited for the wearable and pen based processors recommended in this thesis.

B. JAVA NETWORKING AND DISTRIBUTED COMPUTING.

The Java language has been responsible for making the World Wide Web more than just a colorful billboard, but a real platform that can provide real services. The proliferation of online commerce or "E-commerce" has been exponential and is due in a large part by the Java language's portability and networking capabilities. The potential of the Java language has just now started to catch the eye of professional programmers, as many of the bugs associated with the introduction of a new programming language are discovered and

corrected. The simplicity and elegance of Java has opened programming up to a more diverse population, with varied educational backgrounds. Applications such as networking and multi-tasking that were once the domain of "guru" programmers, are now made simple and straight forward. By utilizing the optimized building blocks incorporated in the JAVA API, novice programmers can create robust and complex network applications and e-commerce solutions. The new programs are less fault prone because of Java's strong typed style and the availability of optimized code from Sun to perform many standard functions.

The Java Networking API supports both stream and datagram sockets. The stream sockets utilize the *Transmission Control Protocol* (TCP), while the datagram sockets utilize the *User Datagram Protocol* (UDP). The Java API also supports Internet Telephony, multicast sockets and other standard networking capabilities. The procedures for invoking the standard functions are well documented and supported [3].

One of the central features of Java is the distributed nature of the *Applet* architecture. Applets allow users to run programs from a remote server, without any client side set up. The software downloaded is only the code required by that application, with most of the code preinstalled in the Java virtual machine. This minimizes the amount of network bandwidth required [2]. The virtual machine can be downloaded from Sun Microsystems, but is usually bundled with Internet browsers.

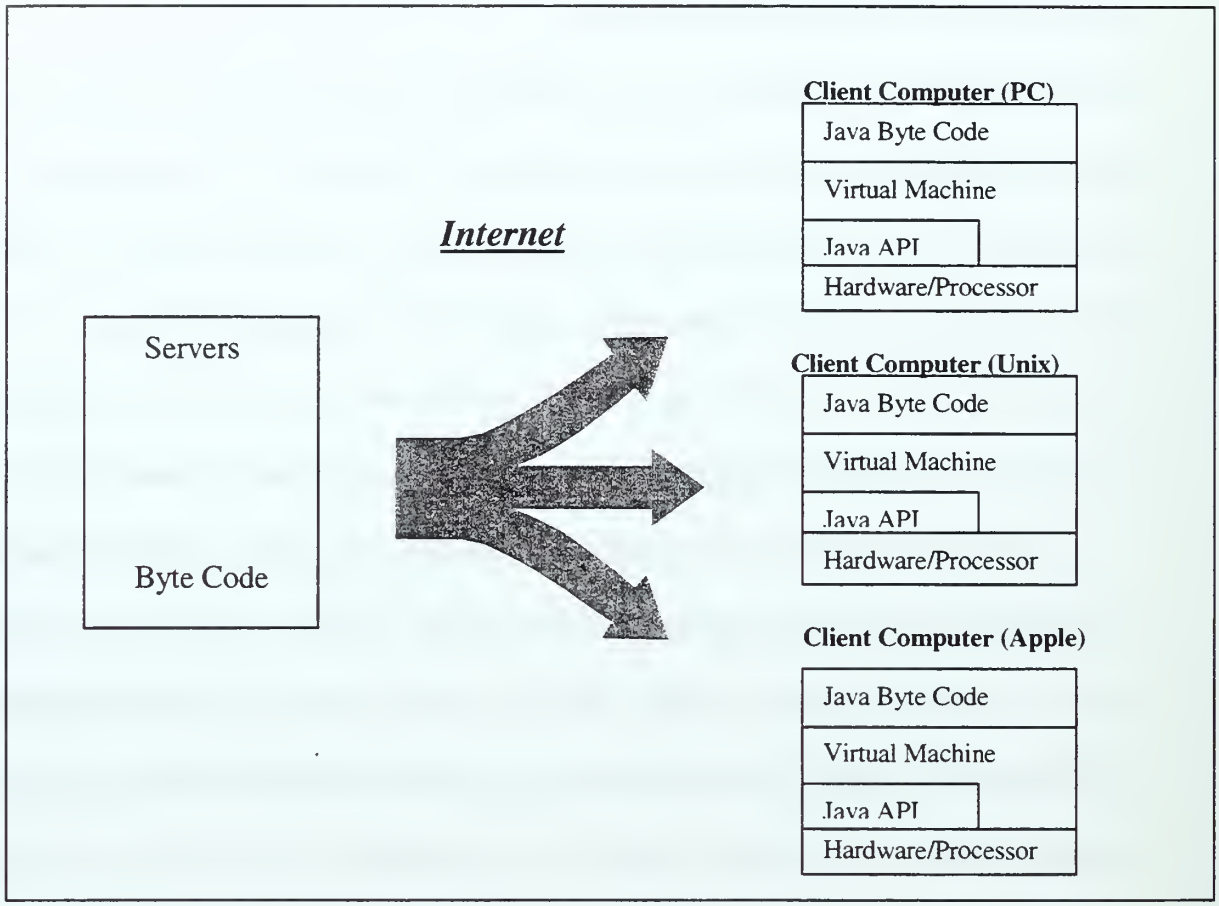

Figure 5. Java Distributed Application Architecture

Java Applet's can perform most of the tasks and computations that other stand-alone languages can, with the exception of some security related restrictions imposed by the language creators, which will be discussed later in this chapter. Some of the significant improvement in the state of computer and networking created by Java's applet architecture stems from its ability to be a portal to a larger processor base from its server, vice running computationally intense applications on the host computer. Java's distributed nature makes

it uniquely suited to interface with various databases and to create a seamless group of databases sharing a similar interface.

In this thesis the prototypes were written in Java under the applet and servlet architecture. The prototype "control station" was written, as an applet, while the "reporting agent" was written as both an applet and a servlet to allow for flexible utilization of available processors and redundancy in case of failure. The applets architecture not only allowed the flexibility normally associated with its distributed nature, but also enhanced the robustness and usability of the application due to its strong typed nature, and installed exception handling, while the servlet architecture allowed less capable devices to interface with the prototype.

C. JAVA MULTIPROCESSOR PLATFORM SUPPORT

With the advent of higher order languages, such as Ada and C++, software developers have sought for a method to write code once for use on a number of different hardware platforms. In the early 1980's, three major operating systems emerged as dominant players serving the three major hardware architectures (Microsoft DOS, supporting the IBM PC architecture, UNIX, supporting various systems and MAC OS, supporting the Apple line of processors). High order languages that supported the creation of software to run on the various operating systems where available, and developers compiled code to run on each operating system, on compilers designed specifically for that hardware/operating system. This method was effective until the early

1990's as most users remained stove piped into their particular operating system and its supplied software base.

In the mid 1990's, the popularity of the World Wide Web drove the industry to search for methods to allow a varying user base to communicate and share applications. Coupled with the significant increase in available network bandwidth and the proliferation of TCP-IP as the de-facto communications standard, Sun Microsystems was able to introduce Java as the premier higher order language to bridge the gap between the three central operating systems over a network [2].

Currently, utilizing the Internet browser as a host, Java's virtual machine is available for a number of operating systems and processor platforms, to include Mac OS, WINTEL, and a number of UNIX releases, and in the near future LINUX. Java has increased the efficiency and availability of software by allowing developers to create the code for an application once for a varied user base, vice creating and managing any number of compiles or builds for the various processor users. As the technology has matured, the number of the inconsistencies between the virtual machines has decreased, permitting the vision of truly portable code to come to fruition.

As the user base of the Internet increases, the number of applications that attract its wide user base with varying levels of platform capabilities increase as well. The user base of the Internet varies from web-TV type platforms running via a slow modem connection to cutting edge multiprocessors attached via a LAN. This user group is turning away from the shrink-wrapped, operating system stove piped software applications to client side Intranet type solutions.

The prototype software created for this thesis attempts to capitalize on the portability of Java, to serve the varied user base in the afloat Navy. By allowing any number of installed computers to run the prototype software, the expense of procuring a new hardware base is avoided. Java's architecture is also uniquely suited to operate on wearable processors, due to the lack of set up and install required by Java (the wearable processors all come with the Java virtual machine preinstalled). The prototype software has been successfully operated on UNIX, PC's and MAC OS, and various hand-held and wearable processors. As industry improves the power and portability of these small systems, they will no doubt continue to support web based applications running in Java.

D. JAVA DATABASE APPLICATIONS.

The Navy continues to operate a huge number of databases. From small text files or Microsoft excel programs, that track command travel expenses to multi-million entry databases such as the pay systems and personnel records system. The systems are stove-piped in nature, with specific client interfaces, that in some cases can only run on propriety processors. The Navy has invested billions of dollars in the data located in these systems and is tied to these legacy databases. Combining all these databases into a current system is a vast undertaking that the DoD could not afford. Java however, offers a partial solution by tying the databases under a common interface.

In order to reach a wider audience throughout the Navy, E-commerce type interfaces written in Java, could access the data located in the legacy databases utilizing

its standard database library. Java supports interface with many databases through its *Structured Query Language* (SQL) library.

E.F. Codd created the Standard Query Language in 1974 at the IBM research laboratory. SQL is designed to retrieve and populate data for relational databases. Most database providers such as Oracle, Microsoft and dBase have adopted the relational database model. SQL was originally intended for use by data processors, which would submit queries to the system in the SQL language. Soon however, higher order

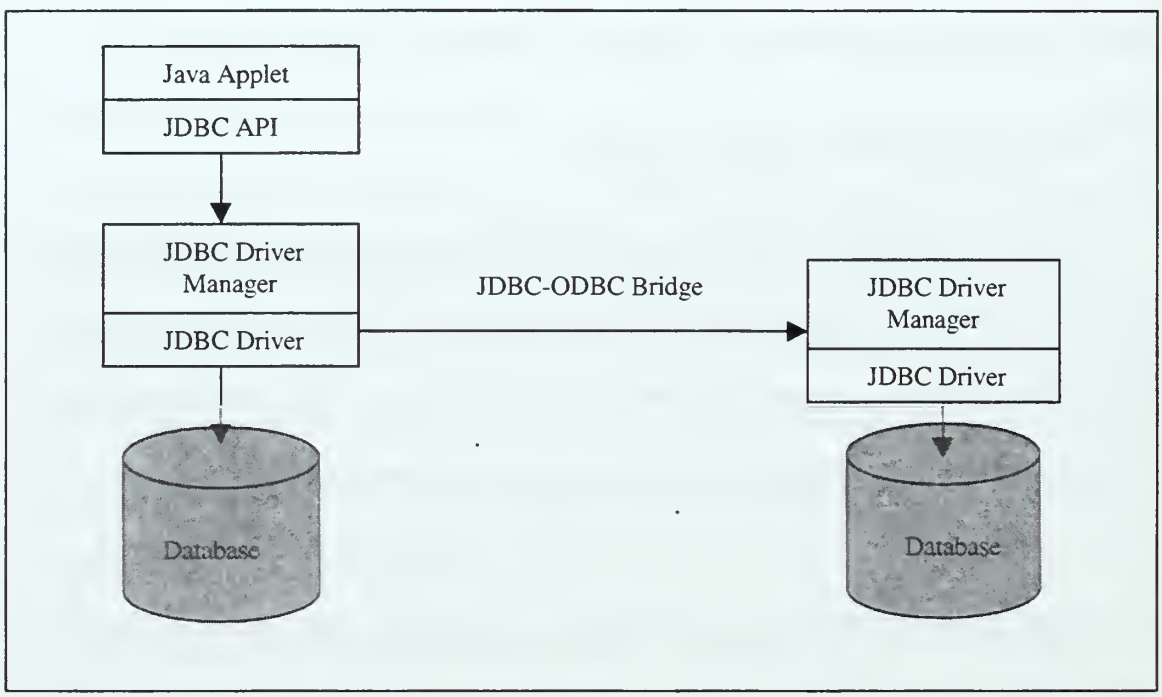

Figure 6. Java Database Architecture

languages with SQL embedded in them where used to create database applications.

To support embedded SQL in Java, Sun Microsystems has developed Java database connectivity (JDBC) [4]. The JDBC API utilizes the 1992 ANSI SQL standard, which many database systems support. JDBC is modeled after Microsoft's ODBC standard, which allows applications to access various data sources. Many database providers have adopted ODBC and Sun Microsystems has introduced a bridge from JDBC to ODBC.

The prototype software discussed in this thesis hopes to leverage the database capabilities of Java to access the various databases in the afloat Navy and tie them together under common interface architecture. In particular the maintenance resource management module endeavors to permit users operating on various processors, including the wearable computer, to access the afloat maintenance database system through a user-friendly graphical interface. The damage control module endeavors to demonstrate the reduction in the complexity of its code, by eliminating the data structure normally associated with that type of application and replacing it with a database flat file.

E. JAVA SECURITY.

One of the greatest concerns surrounding the use of Java distributed systems and open software is security. Reasonable efforts must be made to ensure the integrity and privacy of data. In order to protect users from a malicious applet Sun has placed the following restrictions on Applets:

1. Cannot create a top-level window, unless it's labeled untrusted.
2. Obtain user or home directory name.

3. Access client system hard drive.

4. Create a network connection with any other host than the applet host.

5. Execute a program or spawn a thread outside its thread group.

6. Define system properties.

Java also supports data encryption and verifies byte code prior to execution on the virtual machine. Java's authentication process allows users to designate certain sites as acceptable to perform expanded functions on client machines [3]. As E-commerce becomes more popular, users will become more comfortable trusting personal information to applications running on the Internet, which in turn will allow expanded use of the architecture by the Navy in personnel management applications.

In this thesis, applets will utilize the 128-bit encryption available on most browsers. This level of security is far and away more powerful than any encryption currently utilized to handle non-tactical data. The use of the encryption will ensure that the data transmitted over the wireless LAN will be inaccessible to adversaries. The encryption provided by the browsers has the extended benefits of being constantly scrutinized by an industry that will be held responsible for errors or unintentional disclosure of client information and is available free.

F. CHAPTER SUMMARY

The strength of Java is based in its strongly typed API that finally permits the creation of true object oriented software. By allowing programmers to focus on the "end state" vice the details of implementation, the complexity of software can increase. One of

the reasons C++ and other object oriented languages failed in finally creating a programming environment similar to the building of hardware, is Sun's control of the language specification. When committees or multiple vendor consortiums control a specification the necessity of "pleasing everyone" leads to unnecessarily complicated solutions and slow responses to customer needs. By maintaining strict control over the language, Sun has created a product that is stable enough to encourage wide-spread implementation, yet dynamic enough to embrace new technologies.

One advantage that compiled languages such as C++ or Ada have over a translated language such as Java is speed. Due to the extra steps required by the virtual machine to translate the Java byte code, applications take longer to execute. Sun has taken two approaches to mitigate this delay, the first is the creation of a Java specific microprocessor. Directed at the consumer electronics market the chip has the obvious disadvantage of not supporting other software. Sun has also marketed a "just-in-time" compiler that runs only the required code at the beginning of execution, reducing the total start up time, by spreading the execution time over the entire process. As processor speeds become faster however, Java's advantages in portability, networking and quality outstrip speed for all but the most processor intensive operations.

For the prototype in this thesis, Java provided a superior platform. Distributed in nature, secure by design and portable to a wide-user base, the Java based Intranet provides the Navy with a method to tie together its vast computing resources and information economically and robustly. The Java architecture also serves the wearable and pen based computer base by providing an architecture that capitalizes on the

preinstalled operating systems vice a difficult software client side setup. Ensuring that software will be supported by a wide array of hardware providers, the Navy will no longer be stuck with old software running on slow outdated hardware, but will have the option of improving both economically with industry.

IV. POTENTIAL FOR A WIRELESS SOLUTION

The commercial computer manufacturing industry had provided a solution to aid in increasing the ability of the Navy to truly leverage the duties and responsibilities inherent in the afloat Navy with digital equipment. Advancements in microprocessor and data storage technology have reduced the size of components to allow the incorporation of digital equipment wherever a task is required. Small, lightweight components tied together via the latest generation wireless LAN products have made truly unencumbered use a reality. These computers that utilize pen-based inputs or voice commands have matured sufficiently for reliable use. As most of the wearable computers utilize Intel or Intel compatible processors, they support either Microsoft Windows 95/98/NT or Linux.

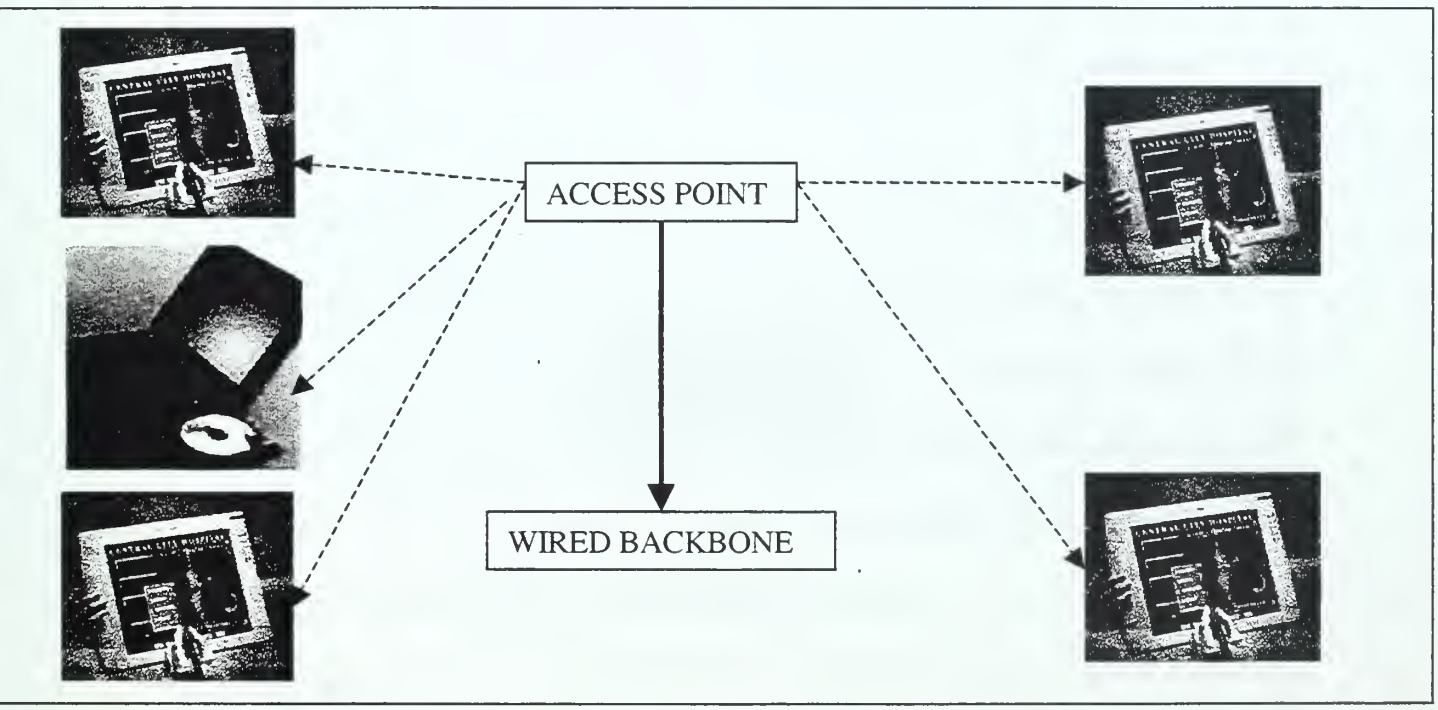

Figure 7. Wireless Local Area Network Architecture

In this thesis the combination of wireless LAN technology and pen or wearable processors operating a Java based Intranet is examined. In the remainder of this chapter, various wireless LAN products and wearable/pen-based processors are examined, with emphases on those devices that best support the mobile nature of the work and the open system paradigm.

A. WIRELESS LAN RADIO EQUIPMENT

Wireless data communications can be divided into four categories based on the desired region of coverage.

1. Personal area Networks (PAN).
2. Local Area Networks (LAN).
3. Metropolitan Area Networks (MAN).
4. Wide Area Networks (WAN).

Wireless Personal Area Networks are designed to provide communications between a user computer and accessory devices such as printers, keyboards and pointing devices. PAN technology consists of both Inferred (IR) or radio technology (RF) to connect devices. The Local Area Network is the traditional type of network that connects computers within a specified area such as an office building. One of the central focuses of the networking industry, the wireless LAN usually utilizes RF technology. The Metropolitan Area Network provides connectivity between numerous LAN and mobile users in a wide region such as a city, town or county. A Wide Area Network combines numerous MAN's and individual

users in a global or national size network. Both WAN's and MAN's utilizes cellular, microwave and satellite technology [6][1]. Since the scale of coverage required for ships and submarines coincides with a Local Area Network (LAN), LAN's will be the focus of this thesis.

The shipboard or submarine environment offers unique challenges in the implementation of wireless LAN technology. The afloat environment in littered with other radio equipment, rotating machinery as well as the metal structure of the platforms itself, which are all hostile to RF devices. The Navy has long awaited the maturity of wireless technology to overcome those challenges to support the true mobility it requires.

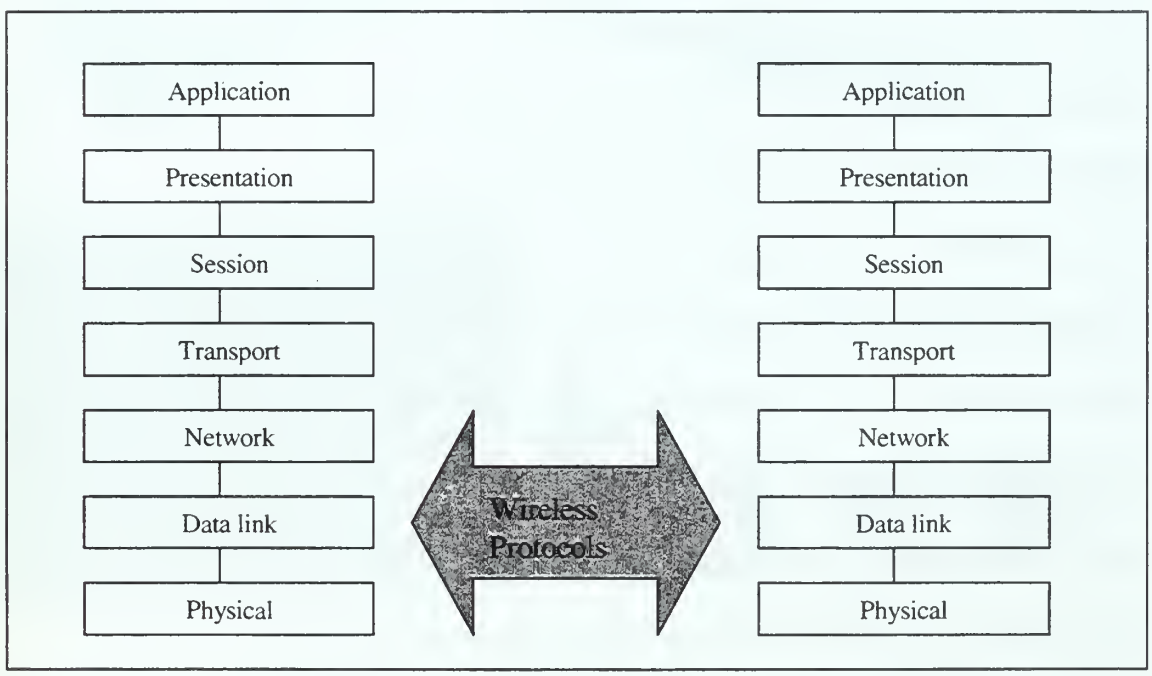

Figure 8. Wireless LAN OSI Model

A Wireless LAN operates in a different manner than traditional wired networks such as Ethernet and TokenRing, to mitigate RF collision difficulties. Wireless LAN technology resides on the bottom three levels of the OSI networking model, and hence requires no changes to the above layers [1].

The wireless LAN utilizes a MAC (Medium Access Control) sublayer protocol known as MACA (Multiple Access with Collision Avoidance). This standard is promulgated in IEEE 802.11, the wireless LAN standard (Tanenbaum, 265). The MACA protocol mitigates the "hidden station problem" and the "exposed station problem". The hidden station problem arises if a wireless LAN is using standard CSMA (Carrier Sense Multiple Access) protocol and a user is transmitting to one unit that is inside its range, while simultaneously another node outside of the original nodes range is also transmitting. Because the original node is unaware of the other transmitting node outside its range, both are transmitting and the frame is lost. The exposed station problem is the opposite condition to hidden station problem, where a node may falsely conclude it cannot transmit due to the transmission of another node to a user outside of its range. The MACA protocol mitigates these problems by sending pre-signals to the nodes prior to transmitting its true cargo. These signal packets basically perform traffic management within the LAN and have dramatically reduced the number of lost frames in a wireless LAN [6].

Various wireless LAN products are available on the commercial market. Most of the new products are IEEE 802.11 compliant and are Direct Sequence Spread Spectrum (DSSS) modulation device that spreads the transmitted information over a wider

bandwidth than required to offset losses due to frequency selective fading experienced in multi-path environments [10]. Operating in the Industrial, Science and Medical (ISM) bands, the wireless local area networking industry supports both 900MHz and 2.4GHz variants.

Most new systems utilize the 2.4 GHz. Bandwidth, transmit at 33 mW, and claim a 2Mbps

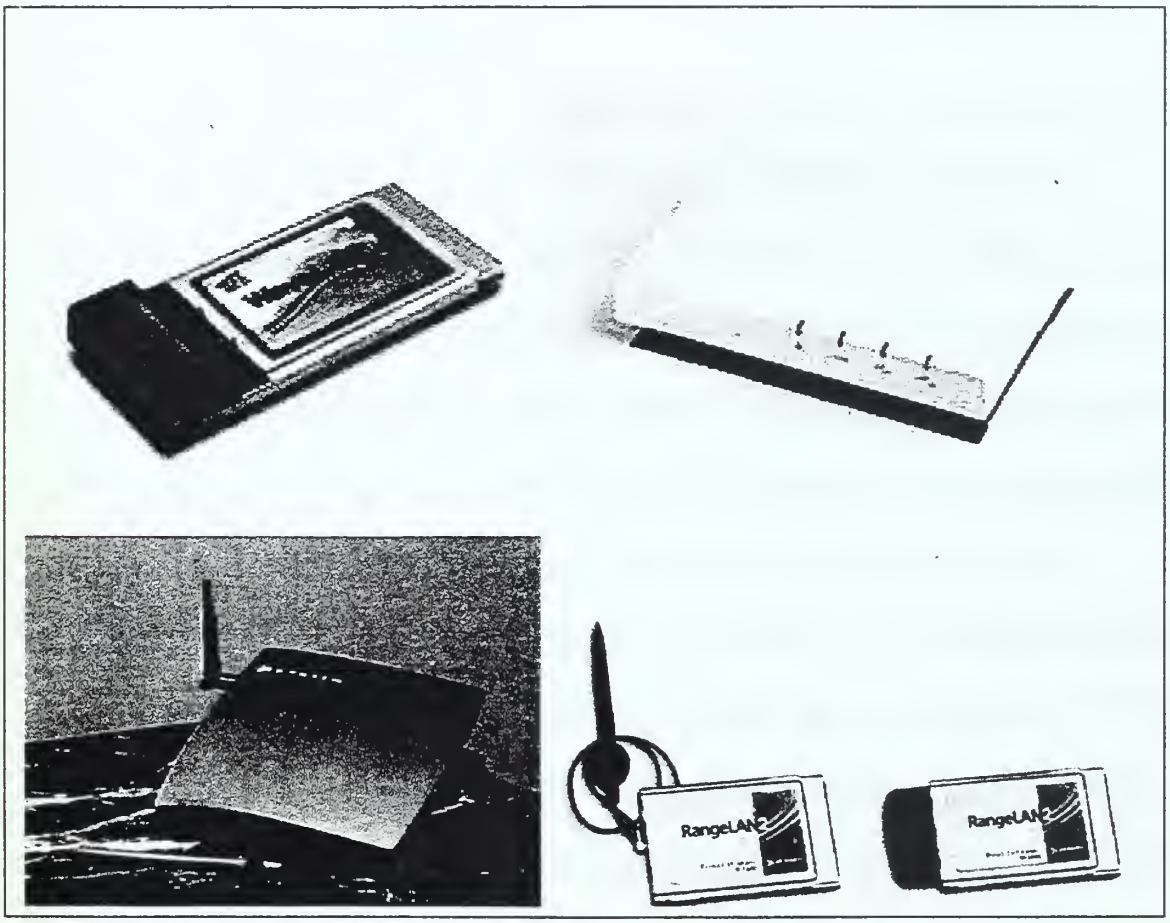

Figure 9. Various Wireless LAN Components

instantaneous data-rate. Many offer the option to either operate in a pier-to-pier mode or from an access point [11]. The benefits of using an access point are easy permanent installation and the ability to route network traffic, allowing the administrator to designate the clients, hence minimizing the amount of broadcast data to those clients. Other benefits of utilizing access points are specific to the hardware manufacturer and can include increased throughput from the access point to the receiver and support for a higher number of client stations.

The systems interface with the client processors through a standard PCMCIA slot, and have available drivers for WINDOWS 3.0/95/NT/CE, MacOS and LINUX. The setup and management are straightforward and follow Simple Network Management Protocols. The systems are designed to support seamless roaming between access points of the same subnet and many have robust backward compatibility.

There are numerous benefits of a wireless LAN over traditional wired LAN. The primary benefit is mobility. The ability to move from one location to another while maintaining seamless connectivity with a host network cannot be overstressed as a work force multiplier. The endless hours wasted in porting data from one platform to another consume energy that could have been better spent. At remote locations, accurate and timely data can improve any number of tasks. Other benefits of a wireless LAN include; reduced initial installation costs, a reduced cost in future upgrades and the ability to install a wireless LAN in areas difficult to wire in a traditional manner. The above benefits become more pronounced as labor increasingly becomes the cost driver in many network installations.

B. MOBILE COMPUTER PLATFORMS

In order to leverage information technology to increase efficiency in the afloat Navy, ubiquitous processors are required for mobility. The computer needs to be available and ergo dynamic when required, as well as unobtrusive. Currently, there are numerous developers that create tailored units and software for heavy industry. Federal Express and Wal-Mart are examples of many companies that utilize these tailored solutions. Many of these developers would be eager to build a similar solution for the Navy, with proprietary hardware and software. However, the commercial market has developed a number or hand-held and wearable computers that run on an open operating system, such as Windows 95/NT and Windows CE [9]. These computers, developed primarily for personal use, are rugged, compact, powerful and support a wide variety of software. By utilizing these products we do not lock ourselves into an outdated product, because of the software investment, as well as capitalizing on the saved costs of research and development that the Navy would certainly have to support. Another added benefit of utilizing COTS equipment, is its ability for expanded use beyond its initial development concept. For example, if the Navy bought a proprietary system, it could only be used for what it was intended for, with any further use, dependent on the developer, however with a COTS device, the users are free to expand the use of the product, beyond its initial vision.

In this section we will examine the attributes of a mobile computer, required by the Navy. A number of processors have been surveyed and recommendations for future purchases and research will be provided.

1. System Requirements

There are a number of attributes that the system must have to support the requirements of the afloat Navy. These attributes include:

Open Operating System: System must support a common operating system such as, Windows 95/NT or CE.

Network Connectivity: Must support constant and reliable network connectivity.

Rugged: The system must be rugged enough for the afloat environment (However, it is important not to stress this too much, for it raises cost. If it's inexpensive enough, you can just replace it).

Battery-life: System must have at least a few hours battery life, with the radio card active. System must also allow for a hot-swap of batteries without re-boot (especially in Windows 95/NT models).

Mobile input method: System must support pen-based input, and either handwriting recognition or voice recognition.

Mobile view method: System must support a portable viewing method of either a head mounted monocle or sunlight viewable screen.

Comfort: The system must be comfortable in both form and function.

Storage: System must have suitable RAM and hard drive capabilities to support full function applications.

Cost: System should be inexpensive, because there is no need for it to be expensive.

By utilizing the above factors, a number of units were surveyed. The units were also examined to determine the difficulty in supporting the prototype software developed in this thesis.

2. Pen-Based Hand Held Units.

Due to the exclusion in this thesis of devices that only support their own proprietary operating systems, such as palm pilots and other vertical market offerings, only hand-held devices that support Windows 95/NT or CE will be discussed. Windows CE is included because it can support the JAVA software, although certain drawbacks are involved when using CE devices, such as its limited ability to support existing Navy WINTEL based software and its minimized Java virtual engine.

Figure 10. CE Operating System Hand Held Device

Pen-based Windows CE units offer some distinct advantages to hand-held Windows 95 and NT units. First, the Windows CE unit has no boot cycle and starts instantly from flash ROM. The lack of boot cycle demonstrates the simplicity and robustness of the operating system. Windows CE also offers superior power management, significantly enhancing battery life. The processors that run Windows CE also tend to consume much less power and are still powerful enough to run complex programs.

Overall, Windows CE devices have a great deal of attributes that make them ideally suited for the type of applications the Navy requires. However, their number one drawback, not supporting a wide range of WINTEL based software, makes the device currently too restrictive. Ideally, the operating system for hand-held computers will operate much like Windows CE, and support all the applications that NT and 95 support.

Another attractive device, suitable for Navy use is the hand-held device utilizing Windows 95 or NT as its operating system. The device shown below has a full color

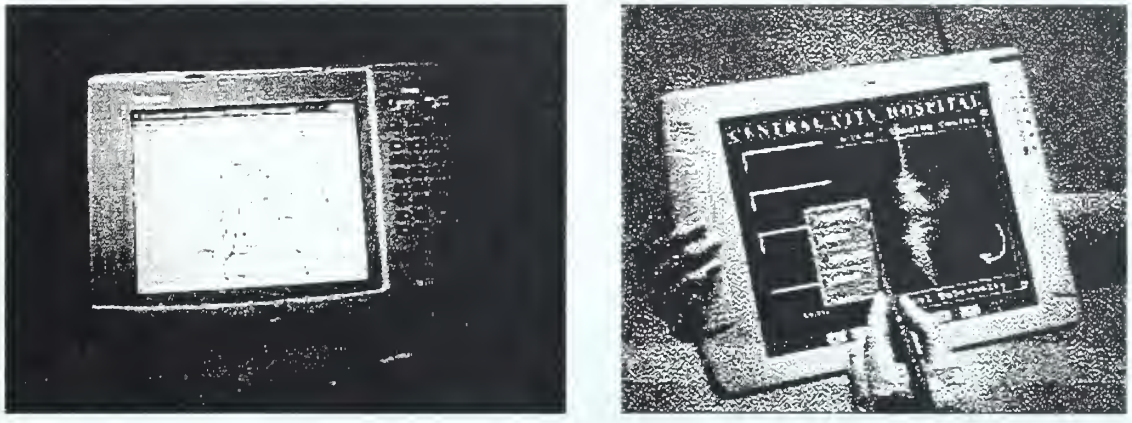

Figure 11. Mitsubishi Amity and Fujitsu Point 510 Windows 95 Hand Held.

screen, runs on Windows 95, and has enough hard drive and RAM to support most applications. The device also supports hand writing recognition and voice commands.

Devices like these are ideal for Navy use; they are rugged, support a wide range of wireless LAN PCMCIA cards and are relatively inexpensive. The Mitsubishi Amity is an excellent device for afloat tasks.

3. Wearable Systems

One of the most important requirements for a mobile computer is comfort. Many of the tasks these systems will be used for are repetitive in nature and will be accomplished in an arduous environment. Hence, any system procured must enhance the task, while not inhibiting mobility. The commercial market has developed two devices that have taken a unique approach to enhancing mobility. By integrating the computer

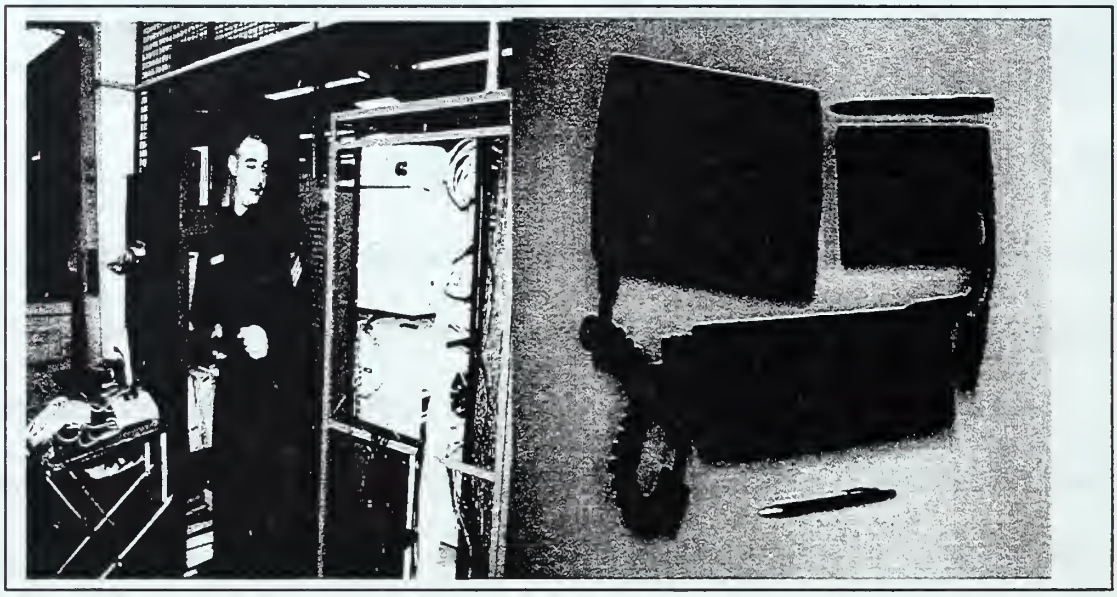

Figure 12. Via's Wearable PC

into a garment or a pack, the user is allowed a full range of motion while still maintaining access to a computer. The first device displayed above was developed by the VIA corporation based in Minnesota.

The device has a belt with three compartments that hold a nickel cadmium battery, the CPU and hard drive, and a holster for the screen. The device is very comfortable and usable, with a bright and readable screen, even in direct sunlight. The hand writing recognition software is good, and the processor speed (180Mhz) is sufficient to operate voice recognition software. The device runs Windows 98, supports the latest Java virtual engine and has excellent power management features. The device utilizes a pressure sensitive stylus to enter information on the screen. This pressure type stylus is preferable

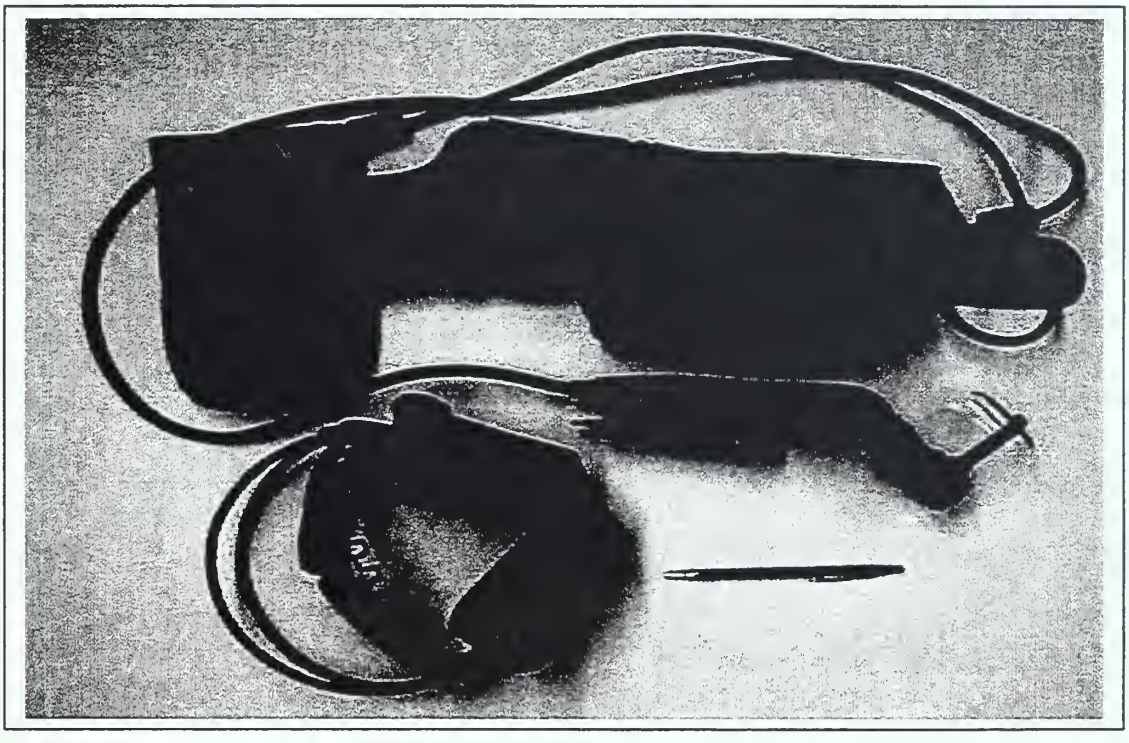

Figure 13. Zybernaut Wearable Device

to other methods, such as magnetic, in that any device can be substituted as a stylus. The device has an excellent battery configuration allowing the user to change batteries without powering down the system.

Overall this device is very suitable for use by the Navy. The "sports like" rugged design, comfortable belt and usability are currently unique in the market. At the reasonable cost of 5000 dollars US each, the device can be procured and fielded to the Navy for a multitude of tasks.

The device shown above, developed by Zybernaut is another wearable device available on the market. This device uses a similar belt configuration as the Flex-PC discussed above, but does not distribute the weight the system over the entire belt, but rather concentrates all the components, minus the battery in one large awkward box. The company offers two types of displays: a monocle or a screen. The monocle offers complete hands free use, and has excellent resolution. The device has a 200Mhz processor, a 2Gb hard drive and runs Window 98.

Overall the Zybernaut is a robust and well-made wearable computer. It lacks, however, some of the central features critical for a mobile platform. These shortcomings include the inability to hot swap the battery, a poor wireless LAN PCMCIA card form factor and a general lack of comfort.

C. CHAPTER CONCLUSIONS

It has been a dream of technology enthusiasts to have a computer that is continuously connected to a network, and is small and durable enough for personal use. The state of technology has finally reached that goal, allowing rich content of various media types to reach the user without a wire. For the Navy, the ability to instantly access timely and accurate data from any location, is a significant force multiplier, that can reap

large saving in training and maintenance. Overall, the combination of wireless LAN and wearable processors is the type of technology that creates solutions to today's problems as well as the future's, that we have not even fathomed.

V. PROTOTYPE SOFTWARE

The purpose of this prototype software is to demonstrate the benefits of utilizing a distributed Java based Intranet as the software architecture for an array of Navy non-tactical software systems. The prototype software is designed to support wearable or hand held computers operating on a wireless LAN, however it can be utilized by any platform with a Java Virtual Machine installed. Most machines have a Java Virtual Machine incorporated into their browser, and hence require no further set up or additional configuration. The concept of the prototype is designed to allow for maximum flexibility, redundancy and economy. Simplicity of code is essential to keep development and maintenance costs down. The overall architecture of the software is shown below.

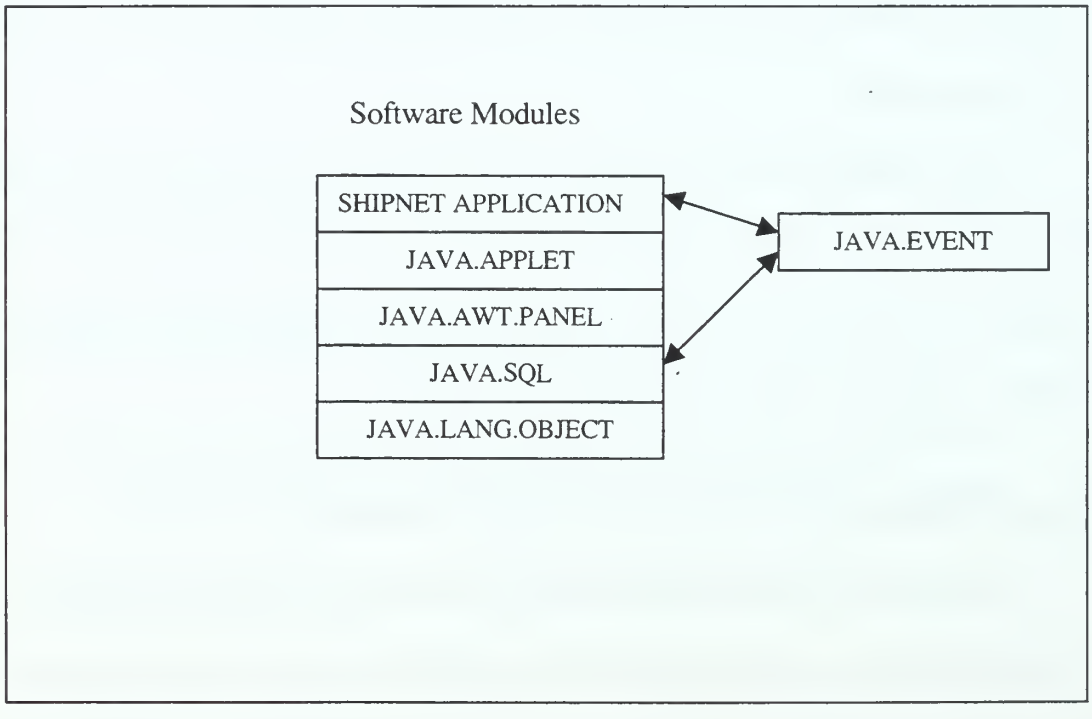

Figure 14. SHIPNET Architecture

Sun Microsystems has already optimized and debugged all the procedures incorporated in the Java Language release, and the Navy can capitalize on that effort. By utilizing Java's internal Intranet Database packages all the socket layer communications has been solved, and hence the focus of the Navy's contractors can then be on creating a usable interface. Furthermore the simplicity in creating the applications allows the Navy to award software contracts to smaller software companies at a dramatically reduced cost.

The maintenance costs associated with the software over its entire lifetime will also be considerably less, when compared to traditional software, due to ease of installation and fielding. Usually, the software has to be installed on each client machine, and is designed to only run on that type of platform. With the Java Intranet architecture any computer on the network can access the software, without any prior configuration, and is installed or updated on the server side only. This simplicity becomes more essential, as more and more items become automated.

In this chapter, the specific structure of the software will be explained. Each module of the prototype is examined individually and the interface created is shown as well. All the modules have been run successfully on Windows NT/95, Unix, Linux, and Macintosh computers. The source code is available in the appendix.

A. APPLET AND SERVLET PROTOTYPE IMPLEMENTATION

As discussed in Chapter II, the benefit of a servlet over an applet is the ability to use HTML exclusively on the client side. This widens the range of client computers considerably, allowing any platform with an HTML browser to act as a client. Numerous

hand-held computers operate light HTML browsers that do not support Java applets, but are ideal candidates for servlets. The client side of the damage control application and the maintenance module of this thesis's prototype have been implemented in both Servlet and Applet forms for comparison. The servlet version can be operated on Windows CE, Apple Newton and other hand held computers with browsers. Applets, however do have some advantages over servlets including; a more robust interface, more distributed computational load (avoids overloading the server, by conducting the bulk of the computation on the client) and finally the fact that, servlets are not supported on all servers. For the prototype software in this thesis the most effective architecture is for the damage control console to be in applet format to support the complex interface, and the client side software to be in servlet format to benefit from the wider range of portable computers. However, as technology advances, and the amount of network bandwidth increases, along with the computing power of hand held devices, the benefits of servlets become less engaging, and the arguments for consistency of the architecture to "all applet" become more substantial.

B. DAMAGE CONTROL APPLICATION

The Damage Control application is the most complex of the modules and incorporates two software applications to perform its function. The application integrates on one Hypertext Markup Language (HTML) page a graphical display of the ship or submarine and a control Java Applet. The graphical image of the ship or submarine is created using the Virtual Reality Modeling Language (VRML). VRML is a programming

language that allows for three-dimensional content to be efficiently downloaded and viewed with an Internet browser. The language is optimized for use in distributed environments such as the Internet or Intranets and provides for animation, sound and interactions, all with a small footprint [5].

VRML images can communicate with JAVA via VRML script nodes or other linkages. JAVA supports a VRML API providing further functionality. The commercial ship and submarine building industry has embraced 3D imaging in new construction as a method to simulate the environment prior to construction to catch design flaws. These same image files can be ported and used in the damage control command console software.

The Damage Control module is divided into two components, a remote or client side and a command and control console. The nature of the software, however, permits the command and control software to be located anywhere a computer is attached to a LAN, whether wireless or traditional.

1. Damage Control Command and Control Module

To support the complex user interface required of the command and control module, the applet architecture was employed. The DC applet utilizes the Java.AWT API to support the graphical user interface, and utilizes the Java.SQL (structured query language) API to interface with a Microsoft Access database. The decision to use of the Microsoft Access database was based primarily on the availability of Access for testing, however any SQL compliant database, such as Oracle, SQL-Server, or DBase could be

used with only a minor change to the configuration of the software. Both the database and the server that maintain the applet code for download reside on the same host. This host computer would most likely be whatever file server the ship or submarine is currently using, however a separate unit running just Windows 98 or NT could be configured to act as the server easily.

In order for applets to access a database from a remote client (not the host computer), a middleware driver must operate between the Java.SQL JDBC and the ODBC Bridge. This type of middleware is not required for Oracle or Microsoft SQL Server databases, but may be required to use the installed database on a ship or submarine and is worth discussing [4]. The middleware sampled in this thesis was developed by IDS Software of Rhode Island, USA. The company is a certified Sun Microsystems vendor, providing a proprietary interface that allows applets to access a database, bundled with server software. The thesis software runs on the IDS server, which negotiates a TCP connection on UDP port 12, which in turn accesses the host database via the IDS driver, declared in the program. A sample of the declarative command is below:

```
import j102.sql.*;
IDSDriver drv = new j102.sql.IDSDriver();
String url = "jdbc:ids://131.120.27.65:12/Maint2";
Connection theConnection = drv.connect(url,null);
```

The import statement allows the applet to see the class j102.sql which is the file with all the proprietary class files. The driver object is instantiated, in this case called

"drv" and then utilized to open a connection utilizing the Java.sql "connect" procedure. In the string names url, the structure is as follows:

String url = "<java driver>:<odcb driver>://<url of host code(codeBase())/<name of databse file>";

A Java.SQL connection object is then instantiated with the driver, the url string. The null entry in connection procedure permits the programmer to enter a username and password if the database is configured to require them.

Once the applet is running and accessing the database, the program interfaces with the database using standard SQL commands. These commands are tied to events in the applets interface. For example in the command console if the "FIRE" button is depressed a SQL command setting the Boolean FIRE field in the database to true for that particular casualty instance, is sent from the applet client through the network to the host server and finally to the database. The command and control applet checks the database every second to see if there is an update, allowing other computers to view the casualty screen while not actually participating in the casualty response. For the shipboard system this configuration is extremely beneficial because it allows other command stations such as combat control, the bridge or surrounding repair stations to remain cognizant of the casualty, just by viewing the applet. There are issues to this open system, like password systems to restrict access or creating a "view only" applet for non-participatory stations, however that is beyond the scope of this thesis.

Below are the interfaces of the damage control console applet. On the top the screen is a sample VRML image created by the author, with the applet running under it.

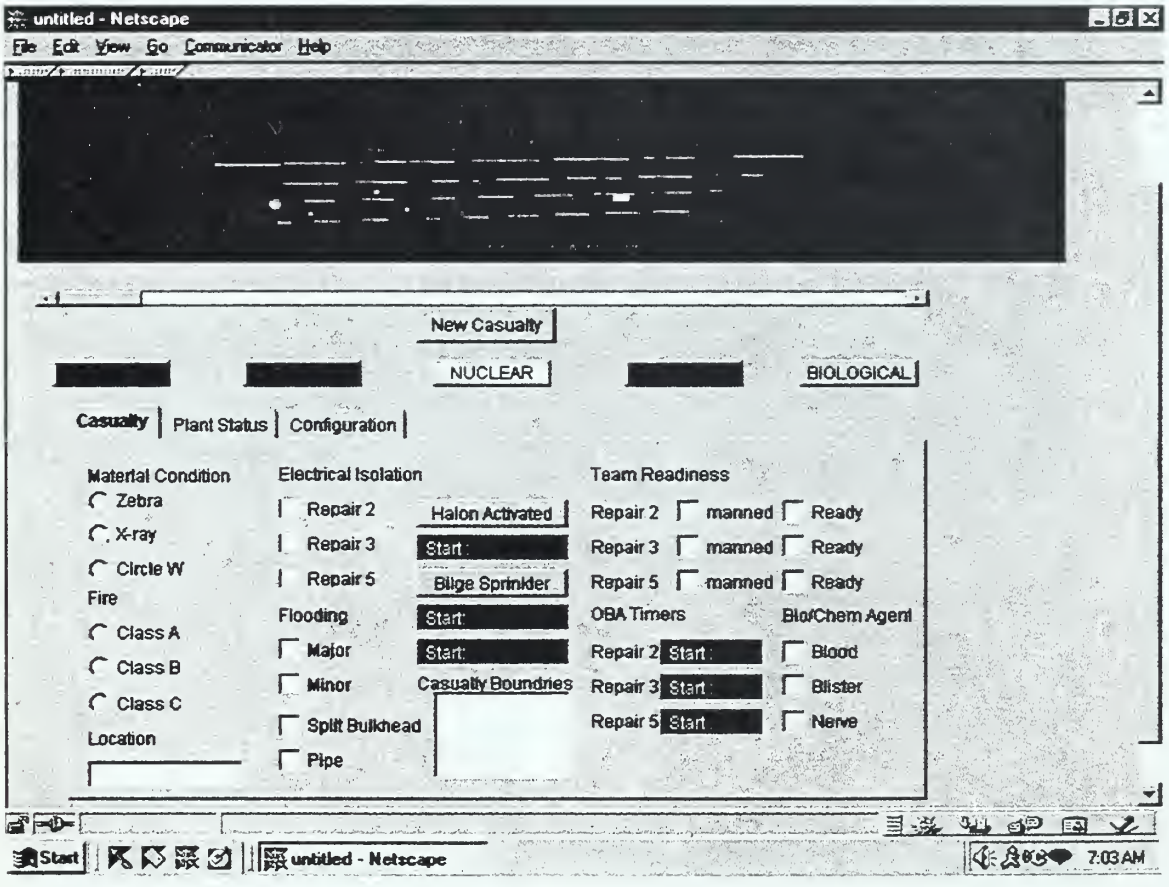

Figure 15. Damage Control Console "Casualty" Tab Panel

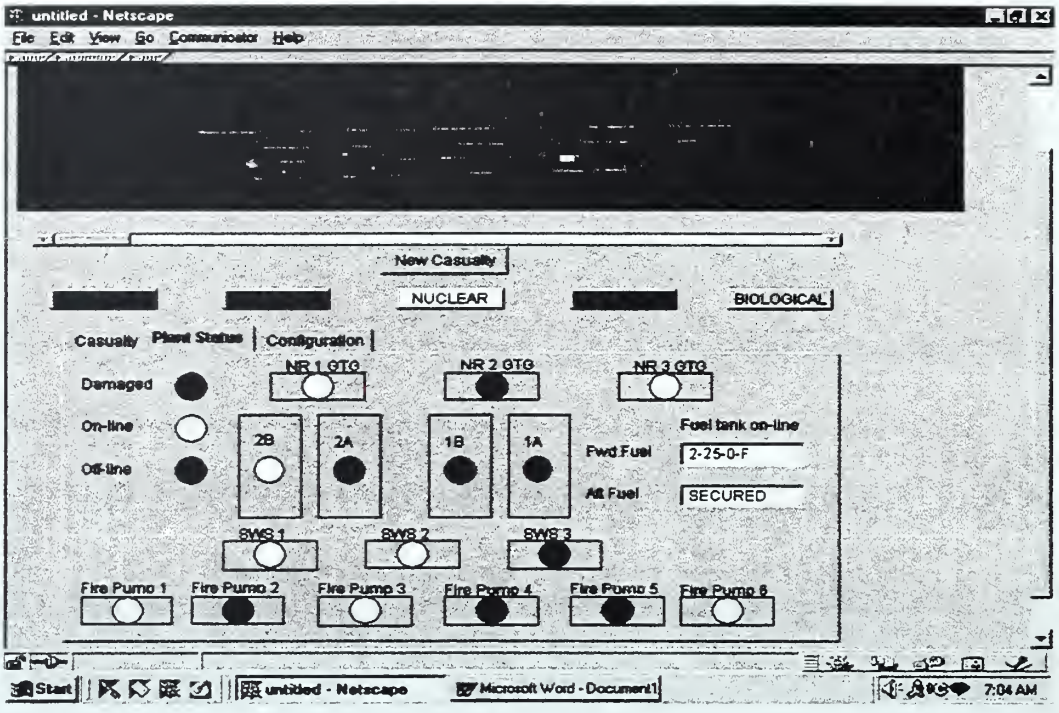

Figure 16. Damage Control Console "Plant Status" Tab Panel

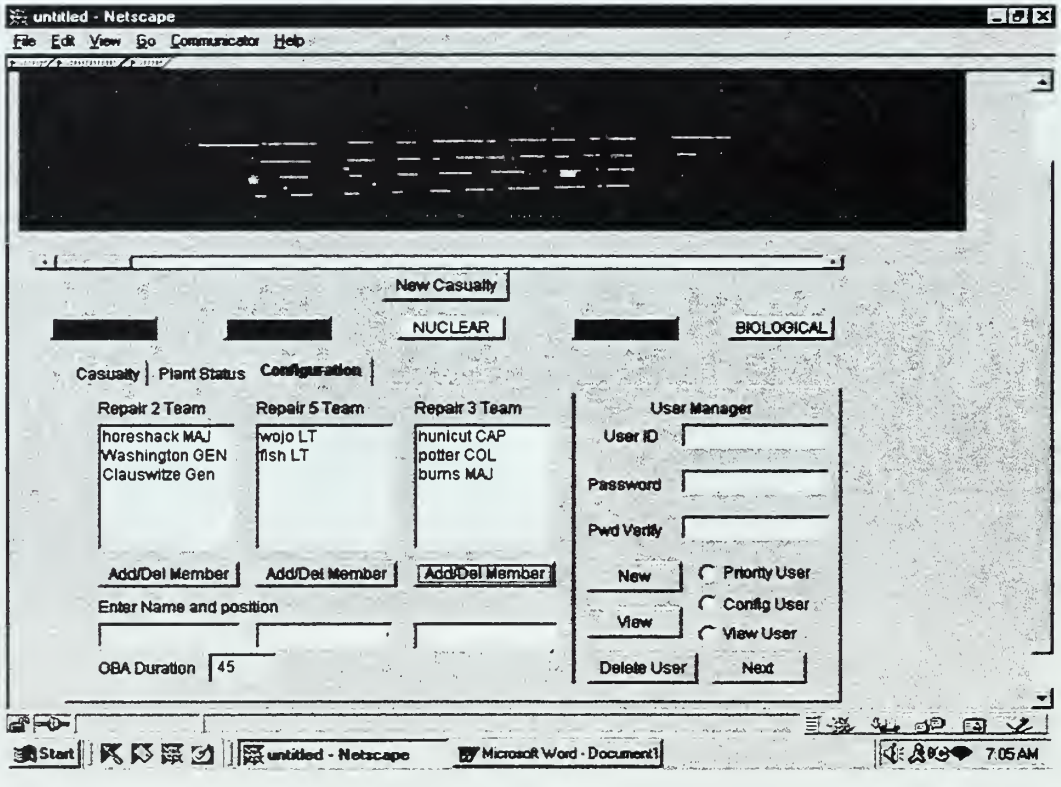

Figure 17. Damage Control Console "Configuration" Tab Panel

In the applet a series of buttons to configure the interface to suit the particular casualty are provided, these are: fire, flooding, nuclear, biological and chemical. As each one of these buttons is depressed the appropriate portions of the interface is enabled. Below the control buttons is a tab panel with three attributes: Casualty, Plant Status and Configuration. The casualty response maintains the pertinent data in a series of text boxes, radio buttons and check boxes. The plant status panel maintains the status of the major engineering and fire fighting equipment, by indicating by color code if the equipment is online, offline or damaged. The panel also informs the user which fuel tanks are online. The last panel contains the configuration details to include the members of the respective repair teams and their positions as well as the oxygen breathing apparatus timer. The panel also has a user manager for future use.

By focusing on utilizing the least amount of "home grown" code, and utilizing as much of Java's API the created code is more likely to be economical and error free. The distributed nature of the application will allow the device to capitalize on the preexisting network and is designed for scalability.

2. Client Module

The client side is a simple response form running as either an applet or servlet depending on the device utilizing it. The interface displayed below is the applet version, which uses the Java AWT to display a series of buttons, textfields and radio buttons. The application interfaces with the host database utilizing the same procedure as the command

application discussed above. In the servlet version, the host code is available on a servlet capable servlet running on the same host as the database. When a client computer requests the servlet, the server activates the servlet code and downloads the HTLM portion of the code to the client. The client is now viewing the interface and can update the status of the casualty. The servlet utilizes the Java.sql to interface with the database when the "submit change" button is depressed. The client side interface updates the readiness of the respective team as well as other items. The same client interface can be utilized by any of the respective repair lockers and run on the wearable and hand-held processors tested in this thesis. The interface was designed to be simple to use and easy to read. The device also was focused on using pen/stylus as the only required means to interface with the application.

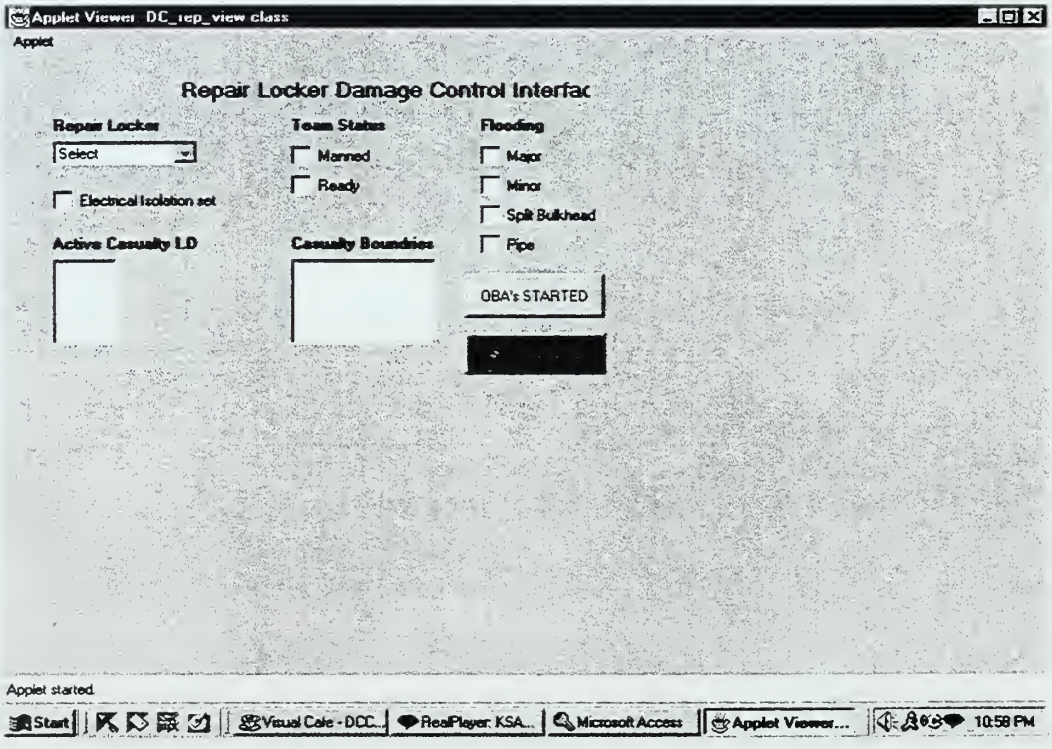

Figure 18. Client Side Damage Control Reporting Interface

C. MAINTENANCE RESOURCE SYSTEM APPLICATION

As discussed in the specification in Chapter II, the maintenance management module allows users to view, edit and create maintenance actions for afloat equipment. For example, if a ship's engine is broken and the problem with the engine requires parts or is beyond the abilities of the crew to fix, a maintenance action must be completed by the ship. Currently, the ship creates this document on a Digital VAX system, that tracks and manages the actions. This software has been ported with limited success to Wintel platforms, but still utilizes the same interface and a client side software installation is required.

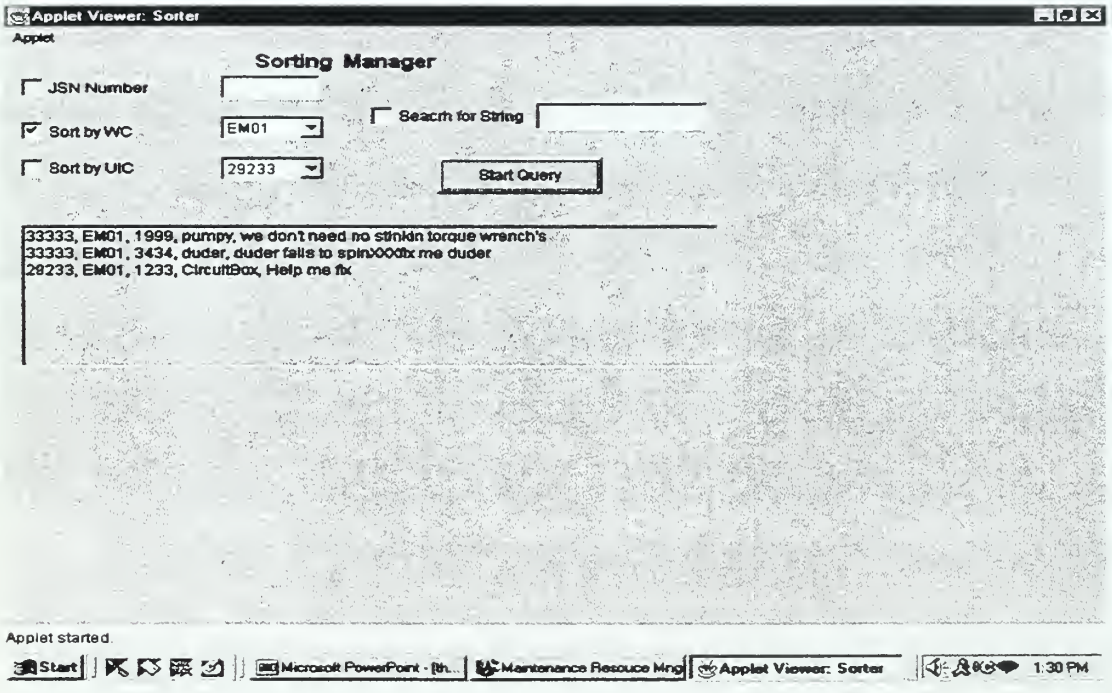

Figure 19. Maintenance Management Interface

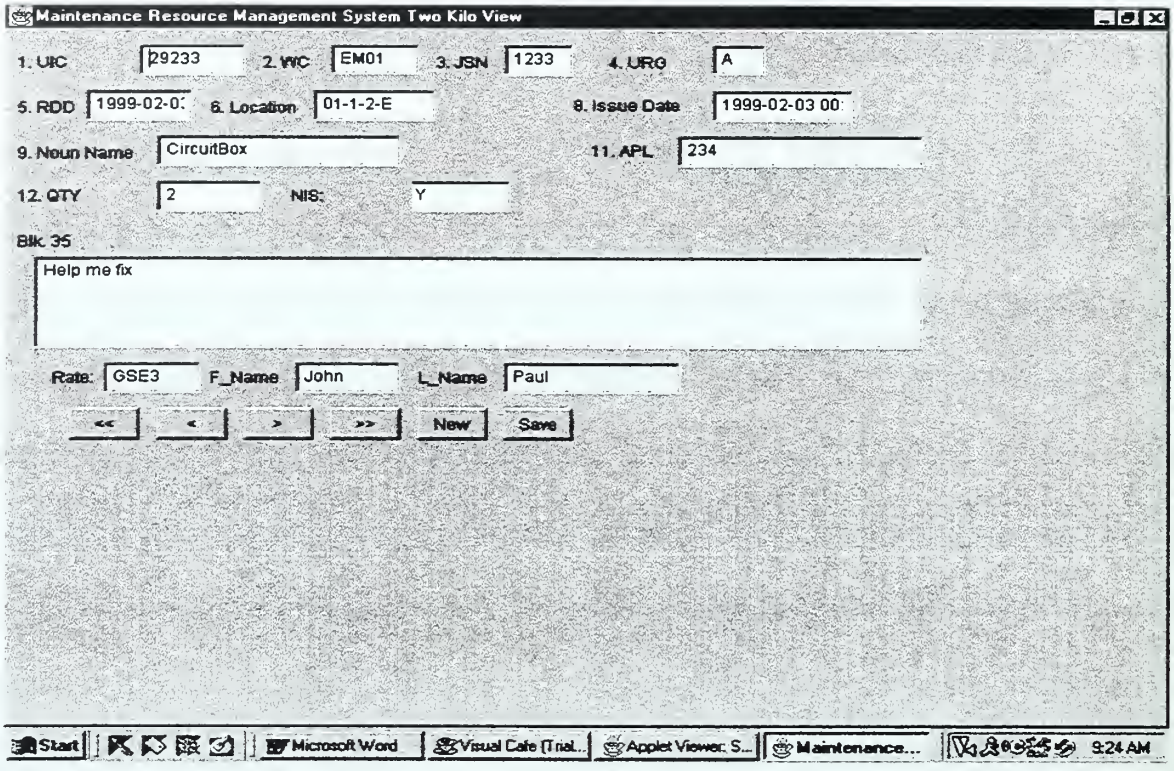

Figure 20. Maintenance Manager New/Edit JSN Interface

The prototype maintenance module in this thesis operates in the same manner that the damage control module does with the exception that the SQL queries and updates in this module are more complicated. The application is created as both a servlet and an applet and can run on any computer with an HTML browser. The benefit of this configuration is the ability to create the maintenance action on the spot, near the equipment, where all the pertinent data resides. This software is also easily extended to provide expert system assists in creating the action, much like a "wizard" assist program. This type of wizard is easy to code, and ensures the maintenance action is complete, with all required information. The client software also has a sorting manager that lets personnel look up jobs by key words, the responsible agent or other fields. This permits the user to look up past jobs and make changes.

Future extensions of the software should combine the supply system, technical manual and maintenance module into one seamless operation that takes the user from problem discovery to troubleshooting then to documentation and parts ordering.

D. CHAPTER SUMMARY

The non-tactical computing technology gap between the afloat Navy and the commercial sector and the Navy shore infrastructure is continuously widening. One of the reasons this gap exists is the mobile nature of the afloat Navy. Unlike an office building, a ship or submarine moves from region to region with varying available bandwidth, hence the shipboard non-tactical systems must transition from full to no connectivity and continue to provide some level of service consistent with the bandwidth limitations. The Navy has numerous programs all going in different directions to provide some computational services. Unfortunately, the applications under development often cannot share data, are usually proprietary in nature and limited in scalability. The method to break the self-fulfilling prophecy is for the Navy to embrace an open system, using proven commercial standards, such as a Java based Intranet. The Navy needs a non-tactical architecture that is a lose confederation of like-minded applications. These applications need to be created in a similar manner, run on any platform, and require no client side configuration, while small enough to run on hand held computers.

The software described in this chapter has all those attributes. The distributed technology and the supporting wireless networking are all mature enough for implementation. The prototype modules demonstrate the robustness currently available in a

distributed computing environment, as well as the usability and economy associated with the architecture.

VI. SHIPBOARD TESTING

In this chapter, the Wireless Local Area Networking and wearable equipment testing conducted on the USS HARRY S. TRUMAN from 30 Mar 99 to 2 APR 99 is documented. Three central technologies were examined during the testing: radio equipment, wearable and pen based processor equipment examination, and a preliminary look at a distributed JAVA based Intranet software application architecture. The testing included an assessment on available bandwidth for various network load environments, a survey of which processor platform was most preferred by crewmembers, and an assessment on acceptability of the "E-commerce" model as a user interface mode.

A. SHIPBOARD TESTING OBJECTIVES AND PROCEDURES

The objectives for the testing was as follows:

1. Determine bandwidth in multi-client environment.
2. Determine number of access points required for NIMITZ CLASS AIRCRAFT CARRIER hanger bay coverage.
3. Determine the most applicable processor platform.
4. Examine software systems to support the wireless/wearable/pen-based architecture.

1. Test Equipment and Software

(2) WAVEPOINT II access points.

(5) WAVEPOINT II PC cards.

(1) XYBERNAUT wearable computer.

(1) VIA II wearable computer.

(1) MITSUBISHI AMITY pen based hand held computer.

(2) LAPTOP computers (one acted as server).

(1) 100 feet of 10BASET cable.

(1) FTP program

(1) Prototype software (Damage Control, Maintenance Management Module).

2. Test Procedures

In order to expedite testing we did not connect the access points to TRUMAN's installed LAN, but created a private network operating from a laptop computer acting as the server. The testing began by measuring throughput with one access point and one client at various ranges. Then the throughput was measured at the same ranges with multiple clients. The throughput measurement was conducted by using an FTP program to simultaneously transfer of a large file (5 MB) from the server to the client and then from the client to the server.

The team also surveyed a number of crewmen on the use of the various processor platforms. The survey attempted to determine which of the four platforms: Xybernaut wearable, notebook laptop, Flex PC wearable, and Mitsubishi Amity hand-held, is most appropriate for afloat use. The use of E-commerce type software was examined as well as the general level of computer familiarity among the survey group was determined.

B. WIRELESS HARDWARE TEST RESULTS

This section discusses the results of the shipboard testing. The wireless LAN equipment testing is presented first, with the software/processor usability study results second. A conclusions and recommendation section follows this section. The graph below demonstrates the results of the throughput in a single access point environment. The graph is a function of throughput in Mbps to range in meters. The decrease in throughput is observed as the range from the access point increases, and the decrease in throughput is relatively linear. The graph shows the throughput for one to three clients and demonstrates

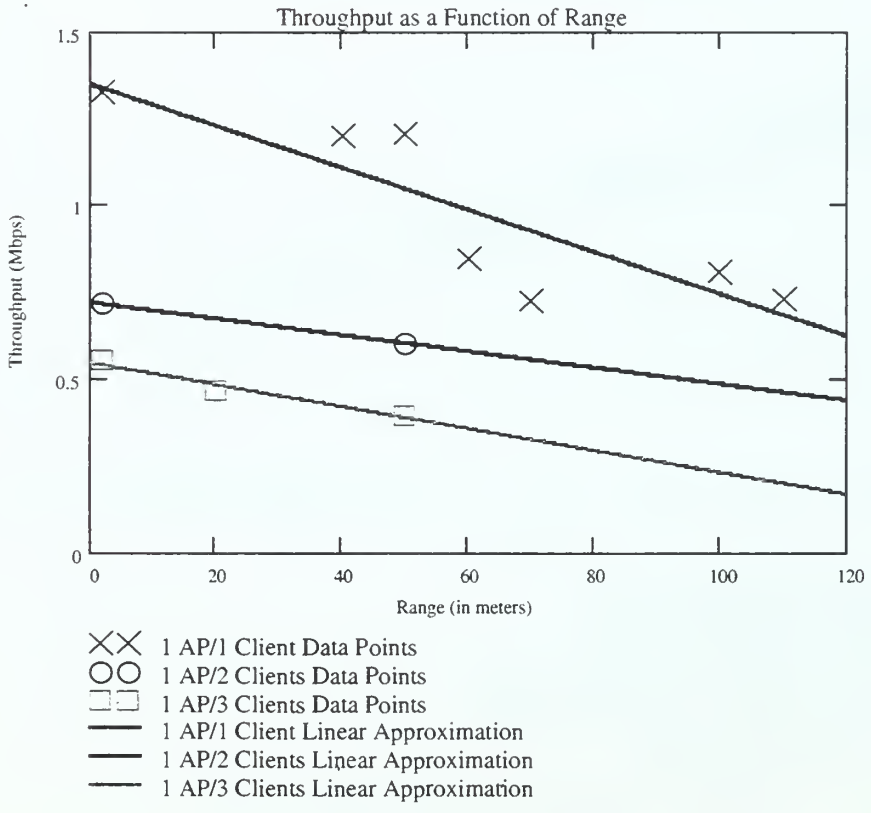

Figure 21. Multi-Client Single Access Point Throughput

how as clients are added the drop in available throughput becomes less dramatic (i.e. the difference between one and two clients is more significant that the difference between two and three).

The graph below is similar to the previous graph except that the cumulative throughput is derived. Demonstrating that the overall throughput is evenly distributed, and that the overhead associated with maintaining the three clients does not effect the

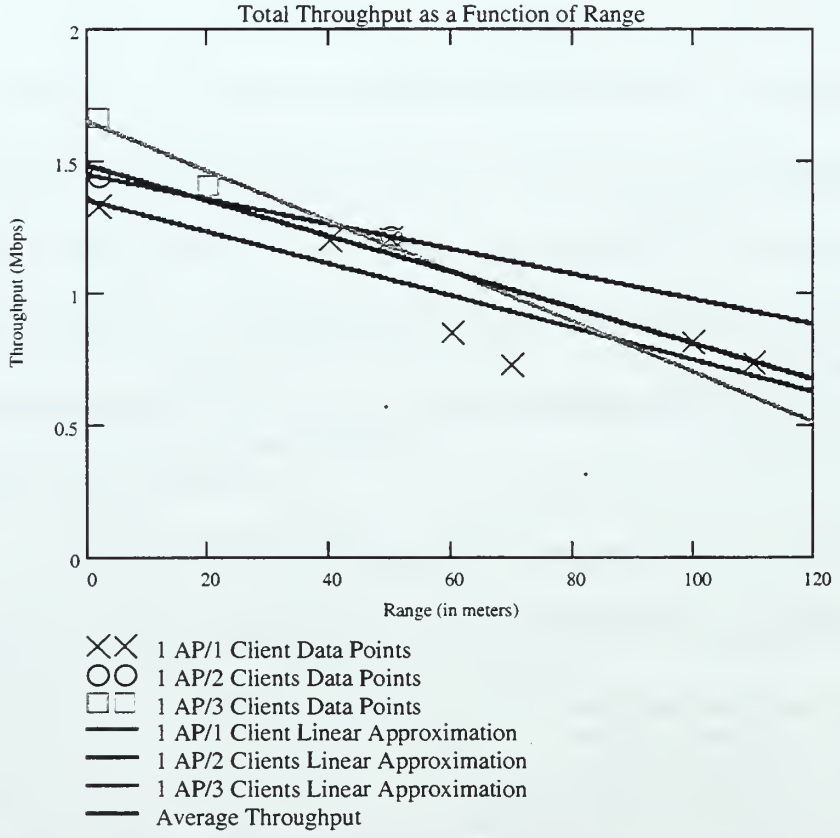

Figure 22. Multi-Client Cumulative Throughput vs. Range

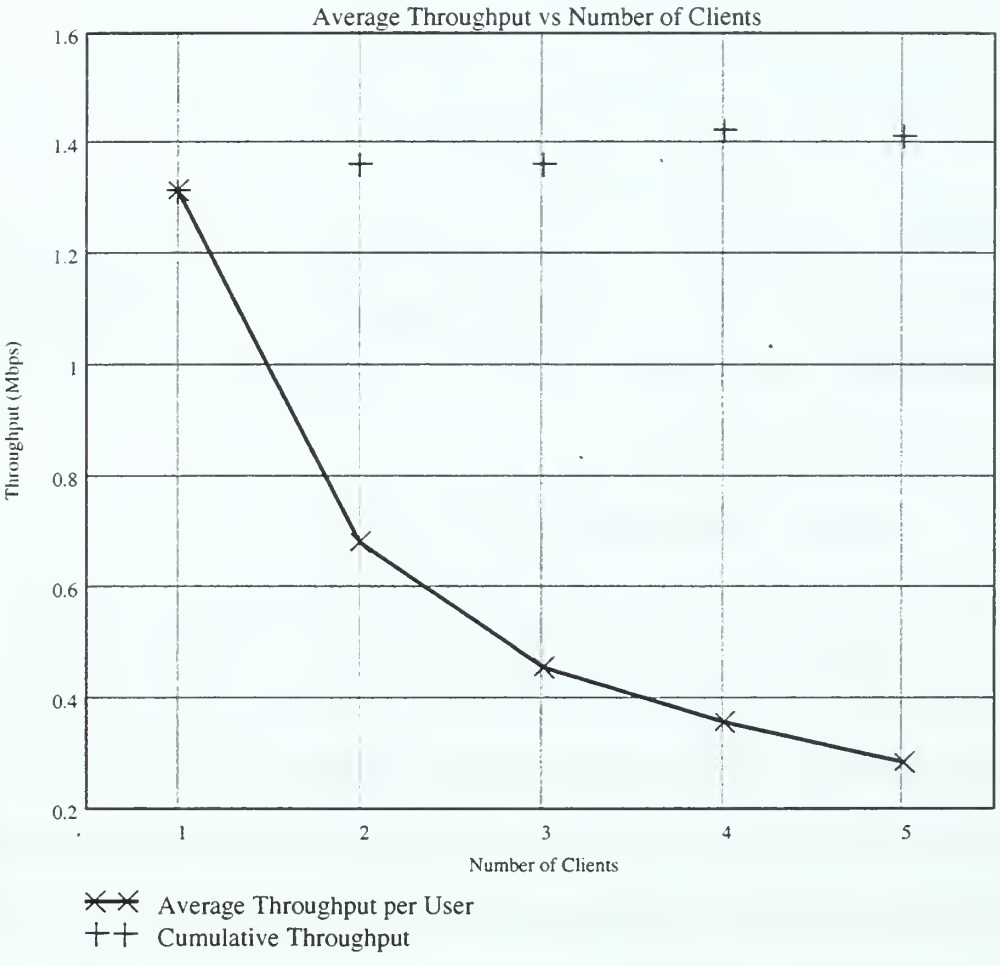

Figure 23. Multi-Client Throughput vs. Range

overall throughput to any great degree. These results demonstrated in this graph appear to support the argument that the overriding concern for bandwidth is the number of clients, vice the range from the access point.

The above graph demonstrates the near exponential decrease in throughput as the number of client's increases. These results were gathered with the clients close and in line of sight to the access points, in a laboratory environment. The clients were at a constant distance from the access point, and executed a near simultaneous file transfer. This graph demonstrates the significant loss of throughput as the number of clients increases and supports the previous assertion that the expected network load should determine the number of access points vice range of coverage.

1. Test One – 1 Access Point with 1 Client

The first access point (AP) was positioned on the starboard side just aft of the hangar bay window at Frame 120. The client was positioned at different locations relative to the access point and a 5 MB file was simultaneously transferred using the file transfer protocol (FTP). The data rate was determined by the aggregate transfer time provided by the FTP server, and verified by taking the file size and evaluating it with the total transfer time. Both the server to client and the client to server data rates where examined. The average data rates from various locations are shown below, the arrows point to the approximate location where data rate measurements were taken.

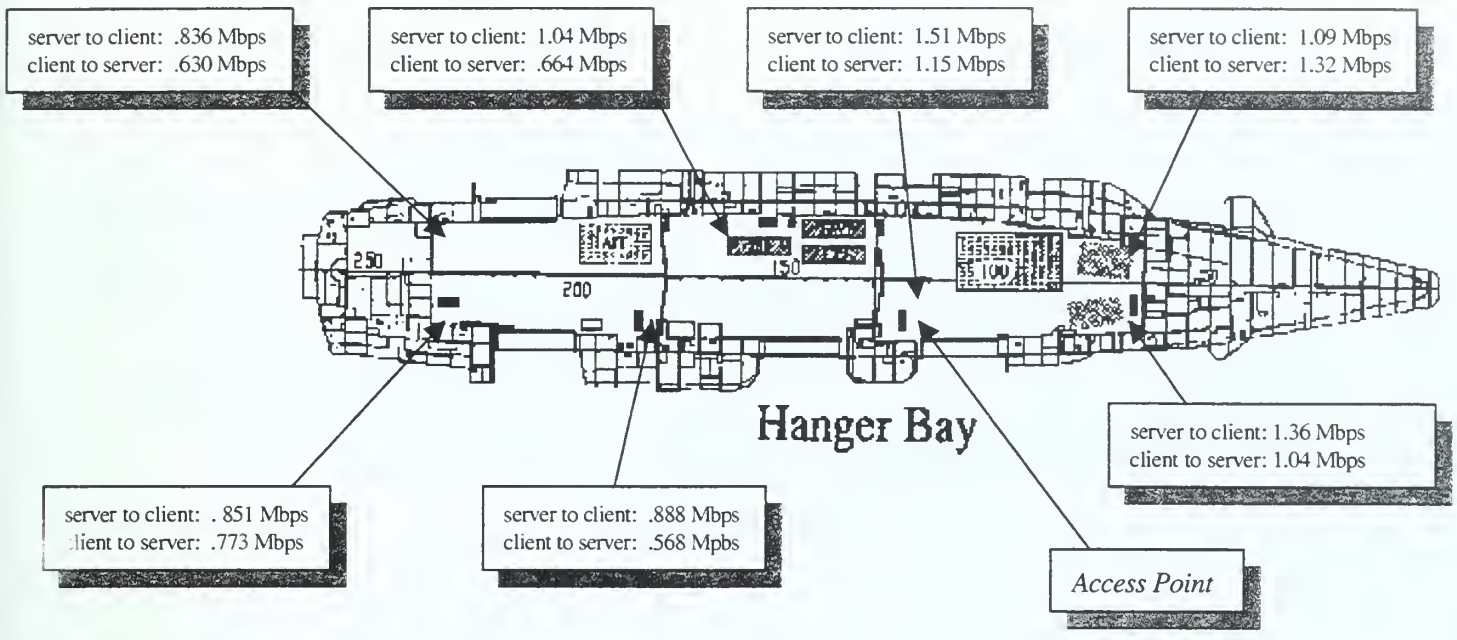

Figure 24. Truman Test One Results

2. Test Two – 1 Access Point with 2 Client (Same Location)

The next test was conducted to examine a multi client environment, to determine any degradation in throughput that might result. The test was conducted by having two clients simultaneously accessing the same access point and conducting an FTP transfer. The flex wearable and amity hand held computers were utilized during the testing. The term "same direction" in the graph below indicates that the computers were both conducting either simultaneous uploads or downloads. The term "Reverse Direction" implies that while one computer was uploading, the other was downloading. All measurements are in Kbps.

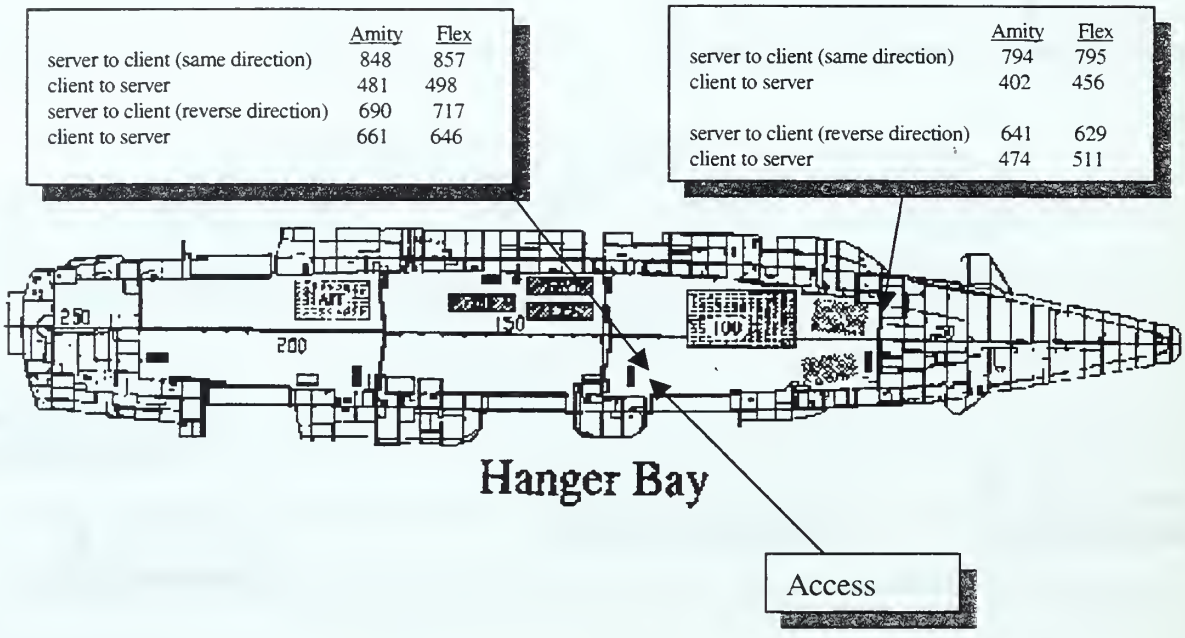

Figure 25. Truman Test Two Results

3. Test Three – 1 Access Point with 3 Clients at Different Locations

The next test was conducted with three clients again conducting simultaneous FTP transfers. For this test the Flex, Amity and Xybernaught were used. The data rate continued to be higher from server to client than from client to server. The unit farthest from the access point suffered the greatest degradation of service, especially when the other two clients where transferring. The tests where conducted in both server to client and client to server mode and configured to determine at the various ranges the effects of uploading while the other two units where downloading.

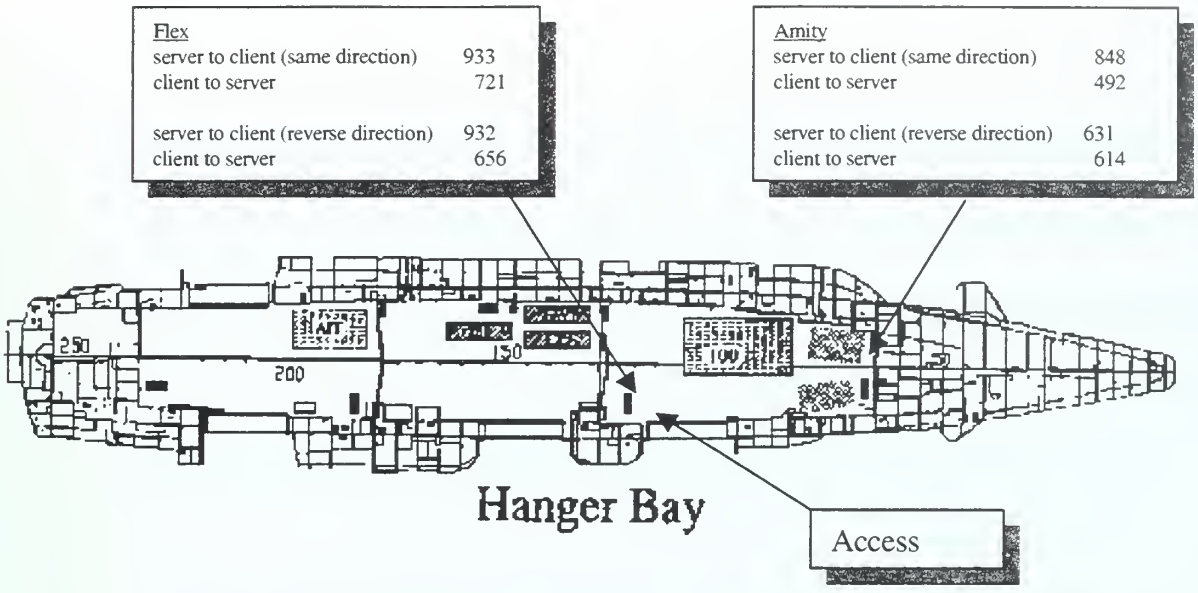

Figure 26. Truman Test Three Results

4. Test Four – 1 Client with 2 Access Points Roaming

Test four examined how seamlessly a client computer could roam between two access points. Two access points where spaced approximately sixty feet apart, and an FTP transfer was initiated was initiated on one access point. The client computer was then moved away from the original access point towards the new access point until the client shifted. No degradation in service or connectivity was noticed during the hand over.

C. SOFTWARE TESTING

During the testing, twelve crewmen of various rates and education were surveyed to determine which applications and processors would be suited for operation on a shipboard wireless LAN. We determined the following:

a. Crewman did not like the headset/monocle style wearable, and preferred the flex PC model.

b. Crewman preferred the writing tablet to virtual keyboards/mini keyboard, voice input.

c. The DC/PMS/TECHMANUAL applications were popular, and additional applications for the device were recommended to include:

Security Patrols

Ships store customer service

Aviation Maintenance Quality Control

A usability study was conducted at NPS and concurred with the TRUMAN results that knowledge of standard human computer interface components were well known at all levels, and hence any hesitation to the Internet (E-commerce) user interface model would be minimal.

D. CONCLUSIONS AND RECOMMENDATIONS

Through both laboratory and shipboard testing the Lucent's WAVELAN IEEE, is recommended as the radio equipment and the VIA flex wearable processor as the processor

platform. The survey also confirmed the desirability for a mobile computing platform for numerous shipboard tasks and the suitability of the JAVA Intranet architecture to support those applications. For the TRUMAN it was interesting to note that the entire hangar bay could be covered by one access point, and that the criteria for wireless LAN installation in an open environment is the number of projected clients.

1. Overall Recommendations

Based on the above testing the following conclusions and recommendations are provided:

1. Wireless Technology has matured enough for integration into the afloat Navy.

2. Lucent's Wavepoint II and supporting equipment are recommended for use due to their overall performance and robust backward compatibility.

3. The Flex PC wearable computer is recommended for its comfortable feel and survivability.

4. The systems should run on either Windows 98 or 95 depending on the ships installed Local Area Network. Although Windows NT is supported, further research is required to determine the difficulty in a NT configuration for the wearables.

5. Since many of the tasks that lend themselves to the mobility provided by wireless technology do not have software written for them, a Java based Intranet type architecture is recommended. Software of this nature is easier to create and maintain

than traditional client-server software and is ideally suited to use with pen based equipment.

2. Further Research

Further research should focus on the following items:

1. Examining other platforms such as submarines and other surface ships.
2. Examining Windows NT issues.
3. Examining security issues.
4. Examining the Java Intranet architecture for additional tasks.

VII. CONCLUSIONS AND RECOMMENDATIONS

The combination of wireless LAN technology and wearable or hand-held computers would greatly improve the efficiency and accuracy of a number of afloat tasks. Alone however, they do not solve the need for economical and tailored software to expand the use of those devices to currently non-automated tasks, where mobile automation is most needed. Today's military program managers and contractors continue to focus on the hardware side of automation, because it is easy to get a handle on. How much RAM is needed? How big a hard drive? Where do we put them? In truth the speed of hardware improvement is so rapid that hardware is virtually disposable. Software is where the life cycle costs of automation reside and where a bad decision in procurement can hamstrung forces for a long time in the future. The Navy continues to utilize stovepipe MS DOS/Wintel based applications with proprietary file formats for fleet wide applications. Due to the sheer size and diversity of the Navy, we must move away from the platform centric to the network centric model, much like an E-commerce structure. Only through utilizing a network centric architecture model will the Navy be able to economically leverage information technology, by reaching every individual and task and gearing towards constant expansion.

A. CONFEDERATION OF LIKE MINDED APPLICATIONS

The Department of Defense has spent billions of dollars on attempting to consolidate non-tactical applications under a common architecture. In one of IBM's most

stunning failures, IBM finished only a few of the hundreds of applications it was contracted to complete and has run way over budget [7]. In 1996 the Navy promulgated IT21, a doctrine designed to set a benchmark for fleet information technology "readiness," and to address the widening non-tactical computing gap between the shore and fleet. The document had the desired effect of mobilizing policy makers to view afloat IT as a higher priority, but it was too directive and precise in its nature to guide the entire Navy's information technology structure.

A new policy that provides general guidance on the architecture of future software implementations is required. The policy should not be directive towards types of systems and parameters, but encourage the creation of a "confederation of like-minded applications." A confederation is preferable to a strict set of procedures because of the vast variety of applications and sheer number of tasks requiring automation. The confederation should have a minimum set of requirements that all future non-tactical applications be Intranet/Internet based depending on scope and require no client side configuration. The software should be able to run seamlessly on a variety of commercial platforms and share a common database if possible, but at a minimum be SQL compliant (no more proprietary storage files). The applications should maintain a consistent look and feel, however this is not critical as the set of standard human computer interface mechanisms are intuitive and familiar enough for a majority of users.

By insisting that all future non-tactical applications be Intranet/Internet based and that all non-volatile storage be SQL compliant databases will ensure that the communications are consistent with the way industry is conducting business and also

allows for easier upgrades and expansion. Today, the Java programming language supports the creation of applications with all of these attributes as well as providing optimized pre-made modules for use in the applications. The prevalence of the E-commerce model in business has the added benefits of lowered costs and support from industries.

B. FORWARD FROM IT-21

The IT-21 project has been centrally responsible for the installation of afloat networks and generally has upgraded the non-tactical processors on surface ships. The instruction describes hardware specifications and operating systems for the ship's computers as well as promulgating the non-tactical network architecture.

The next generation of IT-21 needs to expand to provide further direction for the integration of wireless LAN's and wearable computers. Further more, IT-21 needs to promulgate a general architecture for Navy specific non-tactical software. The architecture of IT-21 compliant software should embrace the attributes discussed in the previous section to include: multi-processor support, no required client side set up, and utilizing a standard communications model such as TCP-IP. These attributes, along with excellent object oriented structure are found in the Java language, which should replace ADA as the standard higher order language for Navy non-tactical software applications.

C. FURTHER RESEARCH

Since the Navy has embraced Windows NT workstation and server as its core administrative computing base, all applications need to be tested for consistent operation in the NT environment. Sun Microsystems and Microsoft continue to have issues regarding Java. The license agreements that allow third party vendors such as Microsoft and Hewlett Packard to create Java virtual machines is strictly enforced by Sun, and those disputes often manifest themselves in subtle differences in how applets and applications work. Further research should continue to examine Java compatibility issues with major operating systems to track the promise of Java as the "write once run anywhere language."

The emergence of server side computational intense distributed computing should also be explored in greater detail. The benefits of placing the bulk of computation on the server lowers the requirements of client processors, thereby allowing less capable systems to still use the software. This approach however places a higher load on the network and server and may reduce the number of clients the server could support. Evaluation comparing server and network utilization between Java Servlets and Applets as the number of client processors increases should be conducted to determine the effective trade off between the two architectures.

Further software prototype applications should be created, to include a cohesive maintenance and supply prototype as well as interactive technical manuals. For the submarine system prototype "rig for dive" applications and equipment log taking

applications should be written. The addition of voice recognition and other interface methods suitable for the hand-held and wearable devices should be examined for compatibility issues with the prototype software, to include new Java API.

D. FINAL CONCLUSIONS

As computers become more pervasive in our society, how we interact with the devices and our expectations will continue to change. The demands to operate more efficiently will continue to increase along with the need for accurate and timely data anytime anywhere. Wearable and hand-held computers as discussed in this thesis offer an attractive way to solve that need for mobile, timely and accurate information and leverage information technology to improve efficiency. Critical to successful improvements, however, is software that allows the Navy flexibility in upgrading and expansion. The combination of wearable processors, wireless LAN equipment and Java-based Intranet software are mature technologies ready now for implementation in the fleet.

APPENDIX A. SOURCE CODE FOR DC CONSOLE APPLICATION

```
// subDC Console screen Applet
// New Attack Submarine Damage Control Applet
// Version 1.1
// Lieutenant Kurt Rothenhaus
// Applet allows the user to update and view DC actions
// for specified US Naval Vessels (Simulated)
// Contains no classified or sensitive data
// Programmer wishes to thank Java; how to program by
// Dietel & Dietel and
// SQL programming in Java by Bill McCarthy for their valuable
// code which I unabashadly used.
// Applet can be viewed at:
// http://131.120.27.65:12/main_DCC_console.htm
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.util.*;
import java.util.Date.*;
import j102.sql.*;
import java.net.*;
import java.io.*;

import symantec.itools.awt.TabPanel;
import symantec.itools.awt.shape.VerticalLine;

public class subDC_consolefin extends Applet
                implements Runnable, ActionListener, ItemListener
{
      public void init()
      {
            // Take out this line if you don't use
symantec.itools.net.RelativeURL or
symantec.itools.awt.util.StatusScroller
            symantec.itools.lang.Context.setApplet(this);


            //{{INIT_CONTROLS
            super.init();
            setLayout(null);
            setSize(426,266);
            setBackground(new Color(12632256));
            new_Cas = new java.awt.Button();
            new_Cas.setLabel("New Casualty");
            new_Cas.setBounds(288,36,96,24);
            new_Cas.setBackground(new Color(12632256));
            new_Cas.addActionListener( this );
            add(new_Cas);
            fire_Btn = new java.awt.Button();
            fire_Btn.setLabel("FIRE");
            fire_Btn.setBounds(156,48,81,18);
            fire_Btn.setBackground(new Color(16711680));
            add(fire_Btn);
            flood_Btn = new java.awt.Button();
            flood_Btn.setLabel("FLOOD");
            flood_Btn.setBounds(444,48,81,18);
```

```
            flood_Btn.setBackground(new Color(255));
            add(flood_Btn);
            horizontalScrollbar1 = new
java.awt.Scrollbar(Scrollbar.HORIZONTAL);
            horizontalScrollbar1.setBounds(12,0,616,11);
            horizontalScrollbar1.setBackground(new Color(16777215));
            add(horizontalScrollbar1);
            tabPanel1 = new symantec.itools.awt.TabPanel();
            try {
                  java.lang.String[] tempString = new
java.lang.String[3];
                  tempString[0] = new java.lang.String("Casualty");
                  tempString[1] = new java.lang.String("Atmosphere
Mon.");
                  tempString[2] = new java.lang.String("Configuration");
                  tabPanel1.setPanelLabels(tempString);
            }
            catch(java.beans.PropertyVetoException e) { }
            try {
                  tabPanel1.setCurrentPanelNdx(2);
            }
            catch(java.beans.PropertyVetoException e) { }
            tabPanel1.setBounds(24,72,612,288);
            add(tabPanel1);
            panel1 = new java.awt.Panel();
            panel1.setLayout(null);
            panel1.setVisible(false);
            panel1.setBounds(12,33,588,244);
            panel1.setBackground(new Color(12632256));
            tabPanel1.add(panel1);
            label1 = new java.awt.Label("Class");
            label1.setBounds(12,15,36,25);
            panel1.add(label1);
            label2 = new java.awt.Label("Fire in:");
            label2.setBounds(108,15,48,25);
            panel1.add(label2);
            label3 = new java.awt.Label("Man in Charge");
            label3.setBounds(12,51,84,29);
            panel1.add(label3);
            label4 = new java.awt.Label("Additional Assit");
            label4.setBounds(12,87,94,20);
            panel1.add(label4);
            PERS = new java.awt.Checkbox("PERS");
            PERS.setBounds(108,87,84,18);
            panel1.add(PERS);
            OBA = new java.awt.Checkbox("OBA");
            OBA.setBounds(192,87,60,20);
            panel1.add(OBA);
            FFE = new java.awt.Checkbox("FFE");
            FFE.setBounds(12,123,75,17);
            panel1.add(FFE);
            DC_EQ = new java.awt.Checkbox("DC_EQ");
            DC_EQ.setBounds(108,123,72,19);
            panel1.add(DC_EQ);
            NFTI = new java.awt.Checkbox("NFTI");
            NFTI.setBounds(192,123,73,19);
            panel1.add(NFTI);
```

```
fire_location = new java.awt.TextField();
fire_location.setBounds(156,15,117,22);
panel1.add(fire_location);
class_fire = new java.awt.TextField();
class_fire.setBounds(60,15,38,20);
class_fire.setText("B");
panel1.add(class_fire);
man_charge = new java.awt.TextField();
man_charge.setBounds(96,51,116,22);
panel1.add(man_charge);
Damaged_equip = new java.awt.TextField();
Damaged_equip.setBounds(108,147,113,23);
panel1.add(Damaged_equip);
Reflash = new java.awt.TextField();
Reflash.setBounds(108,171,114,23);
panel1.add(Reflash);
Hose = new java.awt.TextField();
Hose.setBounds(108,195,113,24);
panel1.add(Hose);
label5 = new java.awt.Label("Damaged Equip");
label5.setBounds(12,147,93,19);
panel1.add(label5);
label6 = new java.awt.Label("Reflash Watch");
label6.setBounds(12,171,94,21);
panel1.add(label6);
label7 = new java.awt.Label("Hose");
label7.setBounds(12,195,94,20);
panel1.add(label7);
NFTI_SAT = new java.awt.Checkbox("NFTI Inspection Sat");
NFTI_SAT.setBounds(12,219,150,17);
panel1.add(NFTI_SAT);
FIRE_IS = new java.awt.TextField();
FIRE_IS.setBounds(408,15,102,21);
panel1.add(FIRE_IS);
label8 = new java.awt.Label("FIRE IS");
label8.setBounds(348,15,48,18);
panel1.add(label8);
label9 = new java.awt.Label("Hose Team A");
label9.setBounds(264,63,84,22);
panel1.add(label9);
label10 = new java.awt.Label("Hose Team B");
label10.setBounds(264,99,84,22);
panel1.add(label10);
label11 = new java.awt.Label("Hose Team C");
label11.setBounds(264,135,84,22);
panel1.add(label11);
label12 = new java.awt.Label("Hose Team D");
label12.setBounds(264,171,84,22);
panel1.add(label12);
Location_A = new java.awt.TextField();
Location_A.setBounds(348,63,84,20);
panel1.add(Location_A);
Location_B = new java.awt.TextField();
Location_B.setBounds(348,99,84,20);
panel1.add(Location_B);
Location_C = new java.awt.TextField();
Location_C.setBounds(348,135,84,20);
```

```
panel1.add(Location_C);
Location_D = new java.awt.TextField();
Location_D.setBounds(348,171,84,20);
panel1.add(Location_D);
HOSE_A = new java.awt.TextField();
HOSE_A.setBounds(444,63,64,20);
HOSE_A.setText("FCML");
panel1.add(HOSE_A);
HOSE_B = new java.awt.TextField();
HOSE_B.setBounds(444,99,64,20);
panel1.add(HOSE_B);
HOSE_C = new java.awt.TextField();
HOSE_C.setBounds(444,135,64,20);
panel1.add(HOSE_C);
HOSE_D = new java.awt.TextField();
HOSE_D.setBounds(444,171,64,20);
panel1.add(HOSE_D);
OBAA = new java.awt.TextField();
OBAA.setText("0:00");
OBAA.setBounds(516,63,64,20);
OBAA.setForeground(new Color(16776960));
OBAA.setBackground(new Color(4210752));
panel1.add(OBAA);
OBAB = new java.awt.TextField();
OBAB.setText("0:00");
OBAB.setBounds(516,99,64,20);
OBAB.setForeground(new Color(16776960));
OBAB.setBackground(new Color(4210752));
panel1.add(OBAB);
OBAC = new java.awt.TextField();
OBAC.setText("0:00");
OBAC.setBounds(516,135,64,20);
OBAC.setForeground(new Color(16776960));
OBAC.setBackground(new Color(4210752));
panel1.add(OBAC);
OBAD = new java.awt.TextField();
OBAD.setText("0:00");
OBAD.setBounds(516,171,64,20);
OBAD.setForeground(new Color(16776960));
OBAD.setBackground(new Color(4210752));
panel1.add(OBAD);
label13 = new java.awt.Label("Location");
label13.setBounds(348,51,80,11);
panel1.add(label13);
label14 = new java.awt.Label("Hose");
label14.setBounds(444,51,63,12);
panel1.add(label14);
label16 = new java.awt.Label("OBA Time");
label16.setBounds(516,51,60,12);
panel1.add(label16);
panel2 = new java.awt.Panel();
panel2.setLayout(null);
panel2.setVisible(false);
panel2.setBounds(12,33,588,244);
panel2.setForeground(new Color(0));
panel2.setBackground(new Color(12632256));
tabPanel1.add(panel2);
```

```
            label15 = new java.awt.Label("O2  (130-220 TORR)");
            label15.setBounds(36,27,120,24);
            label15.setFont(new Font("Dialog", Font.BOLD, 12));
            panel2.add(label15);
            label17 = new java.awt.Label("H2  (27.6 TORR)");
            label17.setBounds(36,63,120,27);
            label17.setFont(new Font("Dialog", Font.BOLD, 12));
            panel2.add(label17);
            label18 = new java.awt.Label("CO  (130-220 TORR)");
            label18.setBounds(36,99,120,28);
            label18.setFont(new Font("Dialog", Font.BOLD, 12));
            panel2.add(label18);
            label19 = new java.awt.Label("CO2  (130-220 TORR)");
            label19.setBounds(36,135,120,30);
            label19.setFont(new Font("Dialog", Font.BOLD, 12));
            panel2.add(label19);
            label20 = new java.awt.Label("R-12/R-114");
            label20.setBounds(36,171,124,32);
            label20.setFont(new Font("Dialog", Font.BOLD, 12));
            panel2.add(label20);
            Group1 = new CheckboxGroup();
            radioButton4 = new java.awt.Checkbox("SAT", Group1, false);
            radioButton4.setBounds(204,27,109,24);
            panel2.add(radioButton4);
            radioButton5 = new java.awt.Checkbox("SAT", Group1, false);
            radioButton5.setBounds(204,63,109,24);
            panel2.add(radioButton5);
            radioButton6 = new java.awt.Checkbox("SAT", Group1, false);
            radioButton6.setBounds(204,99,108,24);
            panel2.add(radioButton6);
            radioButton7 = new java.awt.Checkbox("SAT", Group1, false);
            radioButton7.setBounds(204,135,109,24);
            panel2.add(radioButton7);
            radioButton8 = new java.awt.Checkbox("SAT", Group1, false);
            radioButton8.setBounds(204,171,109,24);
            panel2.add(radioButton8);
            radioButton9 = new java.awt.Checkbox("UNSAT", Group1,
false);
            radioButton9.setBounds(312,27,109,24);
            panel2.add(radioButton9);
            radioButton10 = new java.awt.Checkbox("UNSAT", Group1,
false);
            radioButton10.setBounds(312,63,109,24);
            panel2.add(radioButton10);
            radioButton11 = new java.awt.Checkbox("UNSAT", Group1,
false);
            radioButton11.setBounds(312,99,109,24);
            panel2.add(radioButton11);
            radioButton12 = new java.awt.Checkbox("UNSAT", Group1,
false);
            radioButton12.setBounds(312,135,109,24);
            panel2.add(radioButton12);
            radioButton13 = new java.awt.Checkbox("UNSAT", Group1,
false);
            radioButton13.setBounds(312,171,109,24);
            panel2.add(radioButton13);
            panel3 = new java.awt.Panel();
```

```
panel3.setLayout(null);
panel3.setBounds(12,33,588,244);
tabPanel1.add(panel3);
list2 = new java.awt.List(4);
panel3.add(list2);
list2.setBounds(24,27,105,99);
list2.setBackground(new Color(16777215));
list3 = new java.awt.List(4);
panel3.add(list3);
list3.setBounds(144,27,105,99);
list3.setBackground(new Color(16777215));
list4 = new java.awt.List(4);
panel3.add(list4);
list4.setBounds(264,27,105,99);
list4.setBackground(new Color(16777215));
textField5 = new java.awt.TextField();
textField5.setBounds(24,183,108,20);
panel3.add(textField5);
textField7 = new java.awt.TextField();
textField7.setBounds(144,183,104,21);
panel3.add(textField7);
textField8 = new java.awt.TextField();
textField8.setBounds(264,183,109,22);
panel3.add(textField8);
label33 = new java.awt.Label("Number One Hose");
label33.setBounds(24,3,107,24);
panel3.add(label33);
label34 = new java.awt.Label("Number Two Hose");
label34.setBounds(144,3,107,24);
panel3.add(label34);
label35 = new java.awt.Label("Number Three Hose");
label35.setBounds(264,3,120,24);
panel3.add(label35);
rep2_addmem_btn = new java.awt.Button();
rep2_addmem_btn.setLabel("Add Member");
rep2_addmem_btn.setBounds(24,135,107,20);
rep2_addmem_btn.setBackground(new Color(12632256));
rep2_addmem_btn.addActionListener( this );
panel3.add(rep2_addmem_btn);
rep5_addmem_btn = new java.awt.Button();
rep5_addmem_btn.setLabel("Add Member");
rep5_addmem_btn.setBounds(144,135,107,20);
rep5_addmem_btn.setBackground(new Color(12632256));
rep5_addmem_btn.addActionListener( this );
panel3.add(rep5_addmem_btn);
rep3_addmem_btn = new java.awt.Button();
rep3_addmem_btn.setLabel("Add Member");
rep3_addmem_btn.setBounds(264,135,107,20);
rep3_addmem_btn.setBackground(new Color(12632256));
rep3_addmem_btn.addActionListener( this );
panel3.add(rep3_addmem_btn);
label36 = new java.awt.Label("Enter Name and position");
label36.setBounds(24,159,144,24);
panel3.add(label36);
label37 = new java.awt.Label("OBA Duration");
label37.setBounds(24,207,84,23);
panel3.add(label37);
```

```
            verticalLine1 = new
symantec.itools.awt.shape.VerticalLine();
            verticalLine1.setBounds(384,3,2,228);
            panel3.add(verticalLine1);
            label42 = new java.awt.Label("User Manager");
            label42.setBounds(444,3,84,24);
            panel3.add(label42);
            textField11 = new java.awt.TextField();
            textField11.setBounds(468,63,114,19);
            panel3.add(textField11);
            textField10 = new java.awt.TextField();
            textField10.setBounds(468,27,114,19);
            panel3.add(textField10);
            textField12 = new java.awt.TextField();
            textField12.setBounds(468,99,114,19);
            panel3.add(textField12);
            label43 = new java.awt.Label("User ID");
            label43.setBounds(408,27,48,19);
            panel3.add(label43);
            label44 = new java.awt.Label("Password");
            label44.setBounds(396,63,65,23);
            panel3.add(label44);
            label45 = new java.awt.Label("Pwd Verify");
            label45.setBounds(396,99,70,24);
            panel3.add(label45);
            user = new CheckboxGroup();
            radioButton1 = new java.awt.Checkbox("Priority User", user,
false);
            radioButton1.setBounds(480,135,96,18);
            panel3.add(radioButton1);
            radioButton2 = new java.awt.Checkbox("Config User", user,
false);
            radioButton2.setBounds(480,159,96,19);
            panel3.add(radioButton2);
            radioButton3 = new java.awt.Checkbox("View User", user,
false);
            radioButton3.setBounds(480,183,90,18);
            panel3.add(radioButton3);
            button2 = new java.awt.Button();
            button2.setLabel("New");
            button2.setBounds(396,135,72,24);
            button2.setBackground(new Color(12632256));
            panel3.add(button2);
            button3 = new java.awt.Button();
            button3.setLabel("View");
            button3.setBounds(396,171,72,24);
            button3.setBackground(new Color(12632256));
            panel3.add(button3);
            button4 = new java.awt.Button();
            button4.setLabel("Next ");
            button4.setBounds(492,207,72,24);
            button4.setBackground(new Color(12632256));
            panel3.add(button4);
            button5 = new java.awt.Button();
            button5.setLabel("Delete User");
            button5.setBounds(396,207,84,24);
            button5.setBackground(new Color(12632256));
```

```
        panel3.add(button5);
        textField9 = new java.awt.TextField();
        textField9.setText("45");
        textField9.setBounds(108,207,52,24);
        panel3.add(textField9);

        //}}
    }

    //{{DECLARE_CONTROLS
    JDBC01    theJDBC;        // The object that holds the results
    TextField  theStatus      = new TextField(64);
    String list2del [], list3del [], list4del [];
    Object source1;
  Thread outputThread;
    java.awt.Button new_Cas;
    java.awt.Button fire_Btn;
    java.awt.Button flood_Btn;
    java.awt.Scrollbar horizontalScrollbar1;
    symantec.itools.awt.TabPanel tabPanel1;
    java.awt.Panel panel1;
    java.awt.Label label1;
    java.awt.Label label2;
    java.awt.Label label3;
    java.awt.Label label4;
    java.awt.Checkbox PERS;
    java.awt.Checkbox OBA;
    java.awt.Checkbox FFE;
    java.awt.Checkbox DC_EQ;
    java.awt.Checkbox NFTI;
    java.awt.TextField fire_location;
    java.awt.TextField class_fire;
    java.awt.TextField man_charge;
    java.awt.TextField Damaged_equip;
    java.awt.TextField Reflash;
    java.awt.TextField Hose;
    java.awt.Label label5;
    java.awt.Label label6;
    java.awt.Label label7;
    java.awt.Checkbox NFTI_SAT;
    java.awt.TextField FIRE_IS;
    java.awt.Label label8;
    java.awt.Label label9;
    java.awt.Label label10;
    java.awt.Label label11;
    java.awt.Label label12;
    java.awt.TextField Location_A;
    java.awt.TextField Location_B;
    java.awt.TextField Location_C;
    java.awt.TextField Location_D;
    java.awt.TextField HOSE_A;
    java.awt.TextField HOSE_B;
    java.awt.TextField HOSE_C;
    java.awt.TextField HOSE_D;
    java.awt.TextField OBAA;
    java.awt.TextField OBAB;
    java.awt.TextField OBAC;
```

```
java.awt.TextField OBAD;
java.awt.Label label13;
java.awt.Label label14;
java.awt.Label label16;
java.awt.Panel panel2;
java.awt.Label label15;
java.awt.Label label17;
java.awt.Label label18;
java.awt.Label label19;
java.awt.Label label20;
java.awt.Checkbox radioButton4;
CheckboxGroup Group1;
java.awt.Checkbox radioButton5;
java.awt.Checkbox radioButton6;
java.awt.Checkbox radioButton7;
java.awt.Checkbox radioButton8;
java.awt.Checkbox radioButton9;
java.awt.Checkbox radioButton10;
java.awt.Checkbox radioButton11;
java.awt.Checkbox radioButton12;
java.awt.Checkbox radioButton13;
java.awt.Panel panel3;
java.awt.List list2;
java.awt.List list3;
java.awt.List list4;
java.awt.TextField textField5;
java.awt.TextField textField7;
java.awt.TextField textField8;
java.awt.Label label33;
java.awt.Label label34;
java.awt.Label label35;
java.awt.Button rep2_addmem_btn;
java.awt.Button rep5_addmem_btn;
java.awt.Button rep3_addmem_btn;
java.awt.Label label36;
java.awt.Label label37;
symantec.itools.awt.shape.VerticalLine verticalLine1;
java.awt.Label label42;
java.awt.TextField textField11;
java.awt.TextField textField10;
java.awt.TextField textField12;
java.awt.Label label43;
java.awt.Label label44;
java.awt.Label label45;
java.awt.Checkbox radioButton1;
CheckboxGroup user;
java.awt.Checkbox radioButton2;
java.awt.Checkbox radioButton3;
java.awt.Button button2;
java.awt.Button button3;
java.awt.Button button4;
java.awt.Button button5;
java.awt.TextField textField9;
//}}


public void start( )
```

```
    {
        super.start();
        theJDBC = new JDBC01(theStatus);
        try
        {
            theJDBC.openConnection(); //opens the socket/Bridge
        }
        catch (SQLException sql) { ; }

        try {


                theJDBC.executeQuery("SELECT * FROM rep_2_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list2.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }

      //Opens repair five member list
      try {

                theJDBC.executeQuery("SELECT * FROM rep_5_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list3.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }

      //Opens repair three member list

      try {

                theJDBC.executeQuery("SELECT * FROM rep_3_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list4.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }
            //Start the monitoring thread
            if (outputThread == null) {
            outputThread = new Thread( this );
            outputThread.start();
            }
```

```
    }


public void actionPerformed( ActionEvent e )
          {
          Object source  = e.getSource();

          if  ((source == rep2_addmem_btn) && (source1 == list2del)) {



                try
                   {
               theJDBC.executeUpdate("DELETE FROM rep_2_members WHERE
name_rate = "
                                     + "(" + "'" +
list2.getSelectedItem() + "'" + ")" + ";");


               //textField5.setText("");
                    }
                 catch (SQLException sqlex) { }
               try {

                list2.clear();
                theJDBC.executeQuery("SELECT * FROM rep_2_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list2.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }



               }
            //Just the add

            if  (source == rep2_addmem_btn) {
                try
                    {

               theJDBC.executeUpdate("INSERT INTO rep_2_members VALUES "
                                     + "(" + "'" + textField5.getText()
+ "'" + ")" + ";");

            textField5.setText("");
```

```
                    }
                  catch (SQLException sqlex) { }
                 try {
                 list2.clear();
                 theJDBC.executeQuery("SELECT * FROM rep_2_members;");
                 StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                 while (tokens.hasMoreTokens()){
                   list2.addItem(tokens.nextToken());
                     }
                     }
                   catch (SQLException sql) { ; }



               }


            if  ((source1 == list3del) & (source == rep5_addmem_btn) ) {



                  try
                     {
                theJDBC.executeUpdate("DELETE FROM rep_5_members WHERE
name_rate = "
                                        + "(" + "'" +
list3.getSelectedItem() + "'" + ")" + ";");



                     }
                  catch (SQLException sqlex) { }
               try {

                list3.clear();
                theJDBC.executeQuery("SELECT * FROM rep_5_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list3.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }



            }
```

```
            if  (source == rep5_addmem_btn) {

                try
                    {

               theJDBC.executeUpdate("INSERT INTO rep_5_members VALUES "
                                     + "(" + "'" + textField7.getText()
+ "'" + ")" + ";");


               textField7.setText("");
                    }
                 catch (SQLException sqlex) { }
                try {
                list3.clear();
                theJDBC.executeQuery("SELECT * FROM rep_5_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list3.addItem(tokens.nextToken());
                    }
                    }
                  catch (SQLException sql) { ; }


               }

             if  ((source1 == list4del) & (source == rep3_addmem_btn)) {



                 try
                    {
               theJDBC.executeUpdate("DELETE FROM rep_3_members WHERE
name_rate = "
                                     + "(" + "'" +
list4.getSelectedItem() + "'" + ")" + ";");



                    }
                 catch (SQLException sqlex) { }
               try {

                list4.clear();
                theJDBC.executeQuery("SELECT * FROM rep_3_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list4.addItem(tokens.nextToken());

                  }
```

```
                }
                catch (SQLException sql) { ; }

            }

            if  (source == rep3_addmem_btn) {

               try
                    {

               theJDBC.executeUpdate("INSERT INTO rep_3_members VALUES "
                                     + "(" + "'" + textField8.getText()
+ "'" + ")" + ";");


               textField8.setText("");
                    }
                 catch (SQLException sqlex) { }
                try {
                list4.clear();
                theJDBC.executeQuery("SELECT * FROM rep_3_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list4.addItem(tokens.nextToken());
                    }
                    }
                  catch (SQLException sql) { ; }


               }

        }
      public void run(){
    Thread currentThread = Thread.currentThread();

        while(currentThread == outputThread){
            try{
                showStatus("thread running");
                theJDBC.executeQuery("SELECT * FROM DC;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), ",", false);

                fire_location.setText(tokens.nextToken());
                FIRE_IS.setText(tokens.nextToken());
                man_charge.setText(tokens.nextToken());
                String temp = tokens.nextToken();
                if (temp.equals(" 1"))
                    PERS.setState(true);
                else
                    PERS.setState(false);
                temp = tokens.nextToken();
                if (temp.equals(" 1"))
                    OBA.setState(true);
                else
                    OBA.setState(false);
```

```
                temp = tokens.nextToken();
            if (temp.equals(" 1"))
                FFE.setState(true);
            else
                FFE.setState(false);
            temp = tokens.nextToken();
            if (temp.equals(" 1"))
                DC_EQ.setState(true);
            else
                DC_EQ.setState(false);
            temp = tokens.nextToken();
            if (temp.equals(" 1"))
                NFTI.setState(true);
            else
                NFTI.setState(false);
             Damaged_equip.setText(tokens.nextToken());
             Reflash.setText(tokens.nextToken());
             Hose.setText(tokens.nextToken());
             temp = tokens.nextToken();
            if (temp.equals(" 1"))
                NFTI_SAT.setState(true);
            else
                NFTI_SAT.setState(false);
            //fire_location.setText(tokens.nextToken());
              }
                          catch (SQLException sql) { ; }
             try{
              currentThread.sleep(500);
                }
              catch(InterruptedException e) {}
        }
}


public void itemStateChanged (ItemEvent g)
  {
    source1 = g.getSource();
    list2del = list2.getSelectedItems();
    list3del = list3.getSelectedItems();
    list4del = list4.getSelectedItems();
  }
public void stop( )
    {
        try
        {
            theJDBC.closeConnection( );
            outputThread.stop( );
        }
        catch (SQLException sql) { ; }
    }


}
```

APPENDIX B. SOURCE CODE FOR DC CLIENT APPLICATION

```
// subDC Client Servlet
// New Attack Submarine Damage Control Servlet
// Version 1.1
// Lieutenant Kurt Rothenhaus
// Servlet allows the user to update and view DC actions
// for specified US Naval Vessels (Simulated)
// Contains no classified or sensitive data
// Applet can be viewed at:
// http://131.120.27.65:12/main_DCC_console.htm

import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class DCreturn extends HttpServlet
{


  public void doPost(HttpServletRequest req,
                     HttpServletResponse resp)
    throws ServletException, java.io.IOException
    {
      // Set the content type of the response
      resp.setContentType("text/html");

      // Create a PrintWriter to write the response
      java.io.PrintWriter out =
        new java.io.PrintWriter(resp.getOutputStream());

      // Print the HTML header
      out.println("<html>");
      out.println("<head>");
      out.println("<title>DC Report</title>");
      out.println("</head>");
      out.println("<h2><center>");
      out.println("DC Updated: Press Back button to return");
      out.println("</center></h2>");
      out.println("<br>");

      String values[];

      // Get the location
      String FIRE_IN = "";
      values = req.getParameterValues("Selection");
      if (values != null) {
        FIRE_IN = values[0];
      }
      out.println("FIRE_IN=" + FIRE_IN + "<br>");
      // get state of fire
      String FIRE_IS = "";
      values = req.getParameterValues("Selection2");
      if (values != null) {
        FIRE_IS = values[0];
```

```
 }
 out.println("FIRE_IS=" + FIRE_IS + "<br>");
 //Get man in charge
 String MAN_CHAR = "";
 values = req.getParameterValues("Selection3");
 if (values != null) {
   MAN_CHAR = values[0];
 }
 out.println("MAN_CHAR=" + MAN_CHAR + "<br>");
 // Get the assist
 String PERS = "no";
 String PERS1 = "0";
 values = req.getParameterValues("CheckBox");
 if (values != null) {
   PERS = values[0];
   PERS1 = "1";
 }
 out.println("PERS=" + PERS + "<br>");
 // Get the assist
String OBA = "no";
int OBA1 = 0;
 values = req.getParameterValues("CheckBox13");
 if (values != null) {
   OBA = values[0];
   OBA1 = 1;
 }
 out.println("OBA=" + OBA + "<br>");
 // Get the assist
 String DC_EQ = "no";
 int DC_EQ1 = 0;
 values = req.getParameterValues("CheckBox14");
 if (values != null) {
   DC_EQ = values[0];
   DC_EQ1 = 1;
 }
 out.println("DC_EQ=" + DC_EQ + "<br>");
 // Get the assist
 String NTFI = "no";
 int NTFI1 = 0;
 values = req.getParameterValues("CheckBox15");
 if (values != null) {
   NTFI = values[0];
   NTFI1 = 1;
 }
 out.println("NTFI=" + NTFI + "<br>");
 // Get the assist
 String FFE = "no";
 int FFE1 = 0;
 values = req.getParameterValues("CheckBox16");
 if (values != null) {
   FFE = values[0];
   FFE1 = 1;
 }
 out.println("FFE=" + FFE + "<br>");
 String SAT = "no";
 int SAT1 = 0;
 values = req.getParameterValues("CheckBox17");
```

```
        if (values != null) {
          SAT = values[0];
          SAT1 = 1;
        }
        out.println("SAT=" + SAT + "<br>");
        SAT = String.valueOf(SAT);
        //get damaged equipment
        String DAM_EQU = "";
        values = req.getParameterValues("Selection9");
        if (values != null) {
          DAM_EQU = values[0];
        }
        out.println("DAM_EQU=" + DAM_EQU + "<br>");
        //get watch
        String WATCH = "";
        values = req.getParameterValues("Selection10");
        if (values != null) {
          WATCH = values[0];
        }
        out.println("WATCH=" + WATCH + "<br>");
        //get hose status
        String HOSE = "";
        values = req.getParameterValues("Selection11");
        if (values != null) {
          HOSE = values[0];
        }
        out.println("HOSE=" + HOSE + "<br>");

        try {

              int casualty = 1;
              Connection con = null;
              Statement stmt = null;


              Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();

              con =
DriverManager.getConnection("jdbc:odbc:Maint2",null,null);
              stmt = con.createStatement();
              stmt.executeUpdate("UPDATE DC "
              + "SET FIRE_IN =" + "'" + FIRE_IN + "'"
              + ",FIRE_IS =" + "'" + FIRE_IS + "'"
              + ",MAN_CHAR =" + "'" + MAN_CHAR + "'"
              + ",PERS = "    + "'" +    PERS1  + "'"
              + ",OBA = "     + "'" +    OBA1   + "'"
              + ",DC_EQ = "   + "'" +    DC_EQ1 + "'"
              + ",NTFI = "    + "'" +    NTFI1  + "'"
              + ",FFE = "     + "'" +    FFE1   + "'"
              + ",SAT = "     + "'" +    SAT1   + "'"
              + ",DAM_EQU =" + "'" + DAM_EQU + "'"
              + ",WATCH =" + "'" + WATCH + "'"
              + ",HOSE =" + "'" + HOSE + "'"
              + " WHERE CN=1");
            }
         catch (Exception ex) {
          out.println("Exception!");
```

```
            ex.printStackTrace(out);
          }
        // Wrap up
        out.println("</html>");
        out.flush();
        out.close();
      }

}
// DC remote screen Applet
// New Attack Submarine Demo
// Version 1.0
// Kurt Rothenhaus
// Applet allows the user to update a Damage control
// for specified US Naval Vessels (Simulated)
// Contains no classified or sensitive data
// Programmer wishes to thank Java; how to program by
// Dietel & Dietel and
// SQL programming in Java by Bill McCarthy for their valuable
// code which I unabashadly used.
// Applet can be viewed at:
// http://web.nps.navy.mil/~kjrothen

import java.awt.*;
import java.applet.Applet;
import java.util.*;
import java.awt.event.*;
import j102.sql.*;

public class DC_rep_view extends Applet implements ActionListener {

      JDBC01    theJDBC;        // The object that holds the results
      TextField  theStatus      = new TextField(64);
      java.awt.Choice rep_locker_choice;
      java.awt.Label label1;
      java.awt.Label label2;
      java.awt.Checkbox man_chk;
      java.awt.Checkbox red_chk;
      java.awt.Label label3;
      java.awt.Button submit_btn;
      java.awt.Checkbox elec_chk;
      java.awt.Label label4;
      java.awt.Checkbox major_flood_chk;
      java.awt.Checkbox minor_flood_chk;
      java.awt.Checkbox split_chk;
      java.awt.Checkbox pipe_chk;
      java.awt.Button oba_start;
      java.awt.List list1;
      java.awt.List list2;
      java.awt.Label label5;
      java.awt.Label label6;



      public void init()
      {
```

```
            setLayout(null);
            setSize(426,266);
            setBackground(new Color(12632256));
            rep_locker_choice = new java.awt.Choice();
            rep_locker_choice.addItem("Select");
            rep_locker_choice.addItem("Repair 5");
            rep_locker_choice.addItem("Repair 2");
            rep_locker_choice.addItem("Repair 3");
            try {
                  rep_locker_choice.select(0);
            }
            catch (IllegalArgumentException e) { }
            add(rep_locker_choice);
            rep_locker_choice.setBounds(36,72,110,17);
            rep_locker_choice.setBackground(new Color(16777215));
            label1 = new java.awt.Label("Repair Locker");
            label1.setBounds(36,48,139,22);
            label1.setFont(new Font("Dialog", Font.BOLD, 12));
            add(label1);
            label2 = new java.awt.Label("Repair Locker Damage Control
Interface");
            label2.setBounds(132,12,312,36);
            label2.setFont(new Font("Dialog", Font.BOLD, 16));
            add(label2);
            man_chk = new java.awt.Checkbox("Manned");
            man_chk.setBounds(216,72,100,24);
            add(man_chk);
            red_chk = new java.awt.Checkbox("Ready");
            red_chk.setBounds(216,96,100,24);
            add(red_chk);
            label3 = new java.awt.Label("Team Status");
            label3.setBounds(216,48,139,22);
            label3.setFont(new Font("Dialog", Font.BOLD, 12));
            add(label3);
            submit_btn = new java.awt.Button();
            submit_btn.setLabel("Submit Update");
            submit_btn.setBounds(348,228,108,33);
            submit_btn.setForeground(new Color(0));
            submit_btn.setBackground(new Color(65280));
            submit_btn.addActionListener( this );
            add(submit_btn);
            elec_chk = new java.awt.Checkbox("Electrical Isolation
set");
            elec_chk.setBounds(36,108,168,24);
            add(elec_chk);
            label4 = new java.awt.Label("Flooding");
            label4.setBounds(360,48,139,22);
            label4.setFont(new Font("Dialog", Font.BOLD, 12));
            add(label4);
            major_flood_chk = new java.awt.Checkbox("Major");
            major_flood_chk.setBounds(360,72,100,24);
            add(major_flood_chk);
            minor_flood_chk = new java.awt.Checkbox("Minor");
            minor_flood_chk.setBounds(360,96,100,24);
            add(minor_flood_chk);
            split_chk = new java.awt.Checkbox("Split Bulkhead");
```

```
            split_chk.setBounds(360,120,108,24);
            add(split_chk);
            pipe_chk = new java.awt.Checkbox("Pipe");
            pipe_chk.setBounds(360,144,100,24);
            add(pipe_chk);
            oba_start = new java.awt.Button();
            oba_start.setLabel("OBA's STARTED");
            oba_start.setBounds(348,180,107,35);
            oba_start.setForeground(new Color(16711680));
            oba_start.setBackground(new Color(16762880));
            oba_start.addActionListener( this );
            add(oba_start);
            list1 = new java.awt.List(0);
            list1.setBounds(216,168,110,69);
            list1.setForeground(new Color(0));
            list1.setBackground(new Color(16777215));
            add(list1);
            label5 = new java.awt.Label("Casualty Boundries");
            label5.setBounds(216,144,139,22);
            label5.setFont(new Font("Dialog", Font.BOLD, 12));
            add(label5);
            label6 = new java.awt.Label("Active Casualty I.D");
            label6.setBounds(36,144,139,22);
            label6.setFont(new Font("Dialog", Font.BOLD, 12));
            add(label6);
            list2 = new java.awt.List(0);
            list2.setBounds(36,168,48,69);
            list2.setForeground(new Color(0));
            list2.setBackground(new Color(16777215));
            add(list2);

    }

    public void start( )
    {
        theJDBC = new JDBC01(theStatus);
        try
        {
            theJDBC.openConnection(); //opens the socket/Bridge

        }
        catch (SQLException sql) { ; }
    }

public void stop( )
    {
        try
        {
            theJDBC.closeConnection( );
        }
        catch (SQLException sql) { ; }
    }


 public void actionPerformed(ActionEvent e)
    {
        Object source  = e.getSource();
```

```
        if ((source == oba_start) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){
            //list1.addItem("obabutton5");

        }

        if ((source == oba_start) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){
            //list1.addItem("obabutton3");

        }

        if ((source == oba_start) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){
            //list1.addItem("obabutton2");

        }

        if (source == submit_btn){

        //REPAIR FIVE SET UP
        //*****************************************************//
        if ((man_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){

             list1.addItem("Test");
             try
               {

            theJDBC.executeUpdate("UPDATE dc_casualty SET rep5_man = "
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

            //list1.addItem("Test");


                          }
               catch (SQLException sql) { ; }
        }



   if ((red_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){

             //list1.addItem("Test");
             try
               {

            theJDBC.executeUpdate("UPDATE dc_casualty SET rep5_ready =
"
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

            //list1.addItem("Test");
```

```
                        }
                catch (SQLException sql) { ; }
        }
       if ((major_flood_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET flood_max = "
                                   +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                        }
                catch (SQLException sql) { ; }
        }
      if ((minor_flood_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET flood_min = "
                                   +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                        }
                catch (SQLException sql) { ; }
        }
        if ((elec_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET elec_rep5 = "
                                   +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                        }
                catch (SQLException sql) { ; }
        }
    if ((split_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){
```

```
              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET split_bulk =
"
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                          }
                catch (SQLException sql) { ; }
        }
    if ((pipe_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 5")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET pipe = "
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                          }
                catch (SQLException sql) { ; }
        }


    //REPAIR THREE SET UP
        //*******************************************************//
        if ((man_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET rep3_man = "
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                          }
                catch (SQLException sql) { ; }
        }
```

```
   if ((red_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET rep3_ready =
"
                                    +   "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                           }
                catch (SQLException sql) { ; }
        }
       if ((major_flood_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET flood_max = "
                                    +   "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                           }
                catch (SQLException sql) { ; }
        }
      if ((minor_flood_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET flood_min = "
                                    +   "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                           }
                catch (SQLException sql) { ; }
        }
        if ((elec_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){

              //list1.addItem("Test");
              try
```

```
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET elec_rep3 = "
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                                }
                catch (SQLException sql) { ; }
        }
    if ((split_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET split_bulk =
"
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                                }
                catch (SQLException sql) { ; }
        }
    if ((pipe_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 3")){

              //list1.addItem("Test");
              try
                {

             theJDBC.executeUpdate("UPDATE dc_casualty SET pipe = "
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

             //list1.addItem("Test");


                                }
                catch (SQLException sql) { ; }
        }


//REPAIR TWO SET UP
        //*******************************************************//
        if ((man_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){

              //list1.addItem("Test");
              try
                {
```

```
            theJDBC.executeUpdate("UPDATE dc_casualty SET rep2_man = "
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

            //list1.addItem("Test");


                          }
               catch (SQLException sql) { ; }
        }


   if ((red_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){

             //list1.addItem("Test");
             try
               {

            theJDBC.executeUpdate("UPDATE dc_casualty SET rep2_ready =
"
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

            //list1.addItem("Test");


                          }
               catch (SQLException sql) { ; }
        }
       if ((major_flood_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){

             //list1.addItem("Test");
             try
               {

            theJDBC.executeUpdate("UPDATE dc_casualty SET flood_max = "
                                    +  "(yes)" + "WHERE casualty_id =
1" + ";");

            //list1.addItem("Test");


                          }
               catch (SQLException sql) { ; }
        }
      if ((minor_flood_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){

             //list1.addItem("Test");
             try
               {

            theJDBC.executeUpdate("UPDATE dc_casualty SET flood_min = "
```

```
                                        +  "(yes)" + "WHERE casualty_id =
1" + ";");

              //list1.addItem("Test");


                                }
                  catch (SQLException sql) { ; }
          }
          if ((elec_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){

               //list1.addItem("Test");
               try
                 {

              theJDBC.executeUpdate("UPDATE dc_casualty SET elec_rep2 = "
                                        +  "(yes)" + "WHERE casualty_id =
1" + ";");

              //list1.addItem("Test");


                                }
                  catch (SQLException sql) { ; }
          }
      if ((split_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){

               //list1.addItem("Test");
               try
                 {

              theJDBC.executeUpdate("UPDATE dc_casualty SET split_bulk =
"
                                        +  "(yes)" + "WHERE casualty_id =
1" + ";");

              //list1.addItem("Test");


                                }
                  catch (SQLException sql) { ; }
          }
      if ((pipe_chk.getState()) &
(rep_locker_choice.getSelectedItem().toString() == "Repair 2")){

               //list1.addItem("Test");
               try
                 {

              theJDBC.executeUpdate("UPDATE dc_casualty SET pipe = "
                                        +  "(yes)" + "WHERE casualty_id =
1" + ";");

              //list1.addItem("Test");
```

```
                        }
                catch (SQLException sql) { ; }
        }
    }
    }
}

// JDB01 CLass
// CS3773 Java as a second Language Final Project
// Version 1.0
// Kurt Rothenhaus
// CLASS automates a number of usefull functions that the sorter
// uses to display its results
// Contains no classified or sensitive data
// Programmer wishes to thank Java; how to program by
// Dietel & Dietel and
// SQL programming in Java by Bill McCarthy for their valuable
// code which I unabashadly used.
// Applet can be viewed at:
// http://web.nps.navy.mil/~kjrothen

import java.awt.*;
import java.awt.event.*;
import java.util.Properties;
import java.net.URL;
//import java.sql.*;
import j102.sql.*;  //Creates abstract bridge for socket/JDBC
import java.applet.*;



public class JDBC01
{


    Connection        theConnection = null; // the JDBC bridge
    DatabaseMetaData  theDBMetaData = null;
    Statement         theStatement  = null;
    ResultSet         theResultSet  = null;
    ResultSetMetaData theMetaData   = null;

    TextField theStatus;

    public JDBC01(TextField status)
    {
        theStatus = status;
    }

    // This procedure opens the "socket" to the ODBC bridge
    public void openConnection()
    throws SQLException
    {


        try
```

```
        {
             //  try {
            //   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

            //Driver theDriver =
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            IDSDriver drv = new j102.sql.IDSDriver(); // New ODBC driver
instansiated
            String url =
"jdbc:ids://131.120.27.56:12/conn?dsn='Maint2'";
            //theConnection = DriverManager.getConnection(url, null,
null);
            Connection theConnection = drv.connect(url,null); //connect
is really a java.sql
               // }
                // catch (Exception e){ }

            //Download the database attributes and create a result set.
            theDBMetaData
              = theConnection.getMetaData( );
            theStatement
              = theConnection.createStatement( );
            theResultSet = null;
            theMetaData = null;
            theStatus.setText("Status: OK");
        }
        catch (SQLException sql)
        {
            handleError(sql);
        }

    }
    // Closes the connection to the database when the program is quit.
    public void closeConnection( )
    throws SQLException
    {
        try
        {
            if (theConnection != null)
                theConnection.close( );
        }
        catch (SQLException sql)  { handleError(sql); }
    }
    // Executes the desired query from a string passed in.
    public void executeQuery(String sql)
    throws SQLException
    {
          if (theResultSet != null)
            theResultSet.close( );
          theResultSet = theStatement.executeQuery(sql);
          if (theResultSet != null)
            theMetaData  = theResultSet.getMetaData( );
    }
    //Alows the user to update the database (used in viewscreen)
    public int executeUpdate(String sql)
    throws SQLException
    {
```

```
        if (theResultSet != null)
          theResultSet.close( );
        theResultSet = null;
        theMetaData = null;
        int result = theStatement.executeUpdate(sql);
        return result;
    }
    // Takes the results of the query and turns it into a long string.
    public String dumpResult( )
    throws SQLException
    {
        String result = "";
        try
        {
            int column_count = theMetaData.getColumnCount( );
            while (theResultSet.next( ))
            {
                boolean first = true;
                for (int i = 1; i <= column_count; i++)
                {
                    if (!first) result += ", ";
                    result += theResultSet.getString(i);
                    first = false;
                }
                result += "\n";
            }
        }
        catch (SQLException sql) { handleError(sql); }
        return result;
    }
    // inserts the fields in order
    String getFieldList(String [ ] fields)
    {
        String result = "(";
        boolean first = true;
        for (int i = 0; i < fields.length; i++)
        {
            if (!first) result += ", ";
            first = false;
            result += fields[i];
        }
        result += ") ";
        return result;
    }

    String getValueList(String [ ] values, boolean [ ] isQuoted)
    {
        String result = "VALUES (";
        boolean first = true;
        for (int i = 0; i < values.length; i++)
        {
            if (!first) result += ", ";
            first = false;
            String value = values[i];
            if (isQuoted[i])
            {
                result += "'";
```

```
                // double any embedded single quotes
                int j;
                while ((j = value.indexOf('\'')) >= 0)
                {
                    if (j > 0)
                    {
                        result += value.substring(0, j);
                    }
                    result += "''";
                    if (value.length( ) > j + 1)
                    {
                        value = value.substring(j + 1);
                    }
                    else
                    {
                        value = "";
                    }
                }
                result += value + "'";
            }
            else
            {
                result += value;
            }
        }
        result += ") ";
        return result;
    }

    String getNonNullString(int col)
    throws SQLException
    {
        return nonNull(theResultSet.getString(col));
    }

    String nonNull(String s)
    {
        if (s != null) return s;
        return "";
    }
    // Handles errors that arrise from SQL misrep.
    public void handleError(Throwable t)
    throws SQLException
    {
        theStatus.setText("Error: " + t.getMessage( ));
        t.printStackTrace( );
        throw new SQLException(t.getMessage( ));
    }
}
// DC Console screen Applet
// New Attack Submarine Damage Control Applet
// Version 1.0
// Lieutenant Kurt Rothenhaus
// Applet allows the user to update and view DC actions
// for specified US Naval Vessels (Simulated)
// Contains no classified or sensitive data
```

```
// Programmer wishes to thank Java; how to program by
// Dietel & Dietel and
// SQL programming in Java by Bill McCarthy for their valuable
// code which I unabashadly used.
// Applet can be viewed at:
// http://131.120.21.67:12/main_DCC_console.htm

import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.util.*;
import java.util.Date.*;
//import java.sql.*;
import j102.sql.*;
import java.net.*;
import java.io.*;


import symantec.itools.awt.TabPanel;
import symantec.itools.awt.RadioButtonGroupPanel;
import symantec.itools.awt.ImagePanel;
import symantec.itools.awt.HorizontalSlider;
import symantec.itools.awt.ScrollingPanel;
import symantec.itools.awt.SplitterPanel;
import symantec.itools.awt.BorderPanel;
import symantec.itools.awt.shape.Square;
import symantec.itools.awt.shape.Rect;
import symantec.itools.awt.shape.Circle;
import symantec.itools.awt.shape.VerticalLine;
import symantec.itools.awt.shape.HorizontalLine;


public class dc_Console extends Applet
             implements ActionListener, ItemListener
{

      public void init()
      {
            // Take out this line if you don't use
symantec.itools.net.RelativeURL or
symantec.itools.awt.util.StatusScroller
            symantec.itools.lang.Context.setApplet(this);

            // This code is automatically generated by Visual Cafe when
you add
            // components to the visual environment. It instantiates and
initializes
            // the components. To modify the code, only use code syntax
that matches
            // what Visual Cafe can generate, or Visual Cafe may be
unable to back
            // parse your Java file into its visual environment.
            //{{INIT_CONTROLS
            setLayout(null);
            setSize(426,266);
            setBackground(new Color(12632256));
```

```
            new_Cas = new java.awt.Button();
            new_Cas.setLabel("New Casualty");
            new_Cas.setBounds(276,12,96,24);
            new_Cas.setBackground(new Color(12632256));
            add(new_Cas);
            new_Cas.addActionListener( this );
            fire_Btn = new java.awt.Button();
            fire_Btn.setLabel("FIRE");
            fire_Btn.setBounds(24,48,81,18);
            fire_Btn.setBackground(new Color(16711680));
            fire_Btn.addActionListener( this );
            add(fire_Btn);

            flood_Btn = new java.awt.Button();
            flood_Btn.setLabel("FLOOD");
            flood_Btn.setBounds(156,48,81,18);
            flood_Btn.setBackground(new Color(255));
            flood_Btn.addActionListener( this );
            add(flood_Btn);

            nuclear_Btn = new java.awt.Button();
            nuclear_Btn.setLabel("NUCLEAR");
            nuclear_Btn.setBounds(288,48,81,18);
            nuclear_Btn.setBackground(new Color(16776960));
            nuclear_Btn.addActionListener( this );
            add(nuclear_Btn);

            chem_btn = new java.awt.Button();
            chem_btn.setLabel("CHEMICAL");
            chem_btn.setBounds(420,48,81,18);
            chem_btn.setBackground(new Color(-16744448));
            chem_btn.addActionListener( this );
            add(chem_btn);

            bio_Btn = new java.awt.Button();
            bio_Btn.setLabel("BIOLOGICAL");
            bio_Btn.setBounds(540,48,81,18);
            bio_Btn.setBackground(new Color(16762880));
            bio_Btn.addActionListener( this );
            add(bio_Btn);

            horizontalScrollbar1 = new
java.awt.Scrollbar(Scrollbar.HORIZONTAL);
            horizontalScrollbar1.setBounds(12,0,616,11);
            horizontalScrollbar1.setBackground(new Color(16777215));
            add(horizontalScrollbar1);
            tabPanel1 = new symantec.itools.awt.TabPanel();
            try {
                  java.lang.String[] tempString = new
java.lang.String[3];
                  tempString[0] = new java.lang.String("Casualty");
                  tempString[1] = new java.lang.String("Plant Status");
                  tempString[2] = new java.lang.String("Configuration");
                  tabPanel1.setPanelLabels(tempString);
            }
            catch(java.beans.PropertyVetoException e) { }
            tabPanel1.setBounds(24,72,612,288);
```

```
            add(tabPanel1);
            panel1 = new java.awt.Panel();
            panel1.setLayout(null);
            panel1.setVisible(false);
            panel1.setBounds(12,33,588,244);
            panel1.setBackground(new Color(12632256));
            tabPanel1.add(panel1);
            material_rbs = new CheckboxGroup();
            x_Material_rb = new java.awt.Checkbox("X-ray", material_rbs,
false);
            x_Material_rb.setBounds(12,51,84,24);
            x_Material_rb.setBackground(new Color(12632256));
            panel1.add(x_Material_rb);
            x_Material_rb.setEnabled(false);
            z_Material_rb = new java.awt.Checkbox("Zebra", material_rbs,
false);
            z_Material_rb.setBounds(12,27,84,26);
            z_Material_rb.setBackground(new Color(12632256));
            panel1.add(z_Material_rb);
            z_Material_rb.setEnabled(false);
            w_Material_rb = new java.awt.Checkbox("Circle W",
material_rbs, false);
            w_Material_rb.setBounds(12,75,84,24);
            w_Material_rb.setBackground(new Color(12632256));
            panel1.add(w_Material_rb);
            w_Material_rb.setEnabled(false);
            label1 = new java.awt.Label("Material Condition");
            label1.setBounds(12,15,108,16);
            panel1.add(label1);
            label1.setEnabled(false);
            label2 = new java.awt.Label("Fire");
            label2.setBounds(12,99,84,16);
            panel1.add(label2);
            label2.setEnabled(false);
            fire_rbs = new CheckboxGroup();
            class_A_rb = new java.awt.Checkbox("Class A", fire_rbs,
false);
            class_A_rb.setBounds(12,123,67,15);
            panel1.add(class_A_rb);
            class_A_rb.setEnabled(false);
            class_B_rb = new java.awt.Checkbox("Class B", fire_rbs,
false);
            class_B_rb.setBounds(12,147,79,14);
            panel1.add(class_B_rb);
            class_B_rb.setEnabled(false);
            class_C_rb = new java.awt.Checkbox("Class C", fire_rbs,
false);
            class_C_rb.setBounds(12,171,85,17);
            panel1.add(class_C_rb);
            class_C_rb.setEnabled(false);
            textField1 = new java.awt.TextField();
            textField1.setBounds(12,219,108,19);
            panel1.add(textField1);
            textField1.setEnabled(false);
            label3 = new java.awt.Label("Location");
            label3.setBounds(12,195,48,18);
            label3.setEnabled(false);
```

```
panel1.add(label3);
label4 = new java.awt.Label("Electrical Isolation");
label4.setBounds(144,15,108,12);
label4.setEnabled(false);
panel1.add(label4);
rep_2_chk = new java.awt.Checkbox("Repair 2");
rep_2_chk.setBounds(144,39,72,12);
panel1.add(rep_2_chk);
rep_2_chk.setEnabled(false);
rep_3_chk = new java.awt.Checkbox("Repair 3");
rep_3_chk.setBounds(144,63,72,12);
panel1.add(rep_3_chk);
rep_3_chk.setEnabled(false);
rep_5_chk = new java.awt.Checkbox("Repair 5");
rep_5_chk.setBounds(144,87,72,12);
panel1.add(rep_5_chk);
rep_5_chk.setEnabled(false);
label5 = new java.awt.Label("Flooding");
label5.setBounds(144,111,48,16);
label5.setEnabled(false);
panel1.add(label5);
flood_maj_chk = new java.awt.Checkbox("Major");
flood_maj_chk.setBounds(144,135,88,14);
panel1.add(flood_maj_chk);
flood_maj_chk.setEnabled(false);
flood_min_chk = new java.awt.Checkbox("Minor");
flood_min_chk.setBounds(144,159,60,14);
panel1.add(flood_min_chk);
flood_min_chk.setEnabled(false);
flood_split_chk = new java.awt.Checkbox("Split Bulkhead");
flood_split_chk.setBounds(144,183,108,24);
panel1.add(flood_split_chk);
flood_split_chk.setEnabled(false);
flood_pipe_chk = new java.awt.Checkbox("Pipe");
flood_pipe_chk.setBounds(144,207,84,24);
panel1.add(flood_pipe_chk);
flood_pipe_chk.setEnabled(false);
list1 = new java.awt.List(4);
panel1.add(list1);
list1.setBounds(252,171,94,60);
list1.setBackground(new Color(16777215));
list1.setEnabled(false);
label6 = new java.awt.Label("Casualty Boundries");
label6.setBounds(240,159,108,12);
label6.setEnabled(false);
panel1.add(label6);
halon_btn = new java.awt.Button();
halon_btn.setLabel("Halon Activated");
halon_btn.setBounds(240,39,104,19);
halon_btn.setBackground(new Color(12632256));
halon_btn.addActionListener(this);
panel1.add(halon_btn);
halon_btn.setEnabled(false);
halon_time = new java.awt.TextField();
halon_time.setEditable(false);
halon_time.setText("Start :");
halon_time.setBounds(240,63,106,19);
```

```
halon_time.setForeground(new Color(16776960));
halon_time.setBackground(new Color(4210752));
panel1.add(halon_time);
halon_time.setEnabled(false);
button1 = new java.awt.Button();
button1.setLabel("Bilge Sprinkler");
button1.setBounds(240,87,104,19);
button1.setBackground(new Color(12632256));
button1.addActionListener( this );
panel1.add(button1);
button1.setEnabled(false);
bilge_start_time = new java.awt.TextField();
bilge_start_time.setEditable(false);
bilge_start_time.setText("");
bilge_start_time.setBounds(240,111,106,19);
bilge_start_time.setForeground(new Color(16776960));
bilge_start_time.setBackground(new Color(4210752));
panel1.add(bilge_start_time);
bilge_start_time.setEnabled(false);
bilge_stop_time = new java.awt.TextField();
bilge_stop_time.setEditable(false);
bilge_stop_time.setText("");
bilge_stop_time.setBounds(240,135,106,19);
bilge_stop_time.setForeground(new Color(16776960));
bilge_stop_time.setBackground(new Color(4210752));
panel1.add(bilge_stop_time);
bilge_stop_time.setEnabled(false);
label7 = new java.awt.Label("Team Readiness");
label7.setBounds(360,15,108,12);
label7.setEnabled(false);
panel1.add(label7);
label8 = new java.awt.Label("Repair 2");
label8.setBounds(360,39,48,17);
label8.setEnabled(false);
panel1.add(label8);
label9 = new java.awt.Label("Repair 3");
label9.setBounds(360,63,48,17);
label9.setEnabled(false);
panel1.add(label9);
label10 = new java.awt.Label("Repair 5");
label10.setBounds(360,87,48,17);
label10.setEnabled(false);
panel1.add(label10);
rep2_man_chk = new java.awt.Checkbox("manned");
rep2_man_chk.setBounds(420,39,72,17);
panel1.add(rep2_man_chk);
rep2_man_chk.setEnabled(false);
rep2_red_chk = new java.awt.Checkbox("Ready");
rep2_red_chk.setBounds(492,39,60,17);
panel1.add(rep2_red_chk);
rep2_red_chk.setEnabled(false);
rep3_man_chk = new java.awt.Checkbox("manned");
rep3_man_chk.setBounds(420,63,72,17);
panel1.add(rep3_man_chk);
rep3_man_chk.setEnabled(false);
rep5_man_chk = new java.awt.Checkbox("manned");
rep5_man_chk.setBounds(420,87,72,17);
```

```
panel1.add(rep5_man_chk);
rep5_man_chk.setEnabled(false);
rep3_red_chk = new java.awt.Checkbox("Ready");
rep3_red_chk.setBounds(492,63,60,17);
panel1.add(rep3_red_chk);
rep3_red_chk.setEnabled(false);
rep5_red_chk = new java.awt.Checkbox("Ready");
rep5_red_chk.setBounds(492,87,60,17);
panel1.add(rep5_red_chk);
rep5_red_chk.setEnabled(false);
label11 = new java.awt.Label("Repair 2");
label11.setBounds(360,135,48,17);
label11.setEnabled(false);
panel1.add(label11);
label12 = new java.awt.Label("Repair 3");
label12.setBounds(360,159,48,17);
label12.setEnabled(false);
panel1.add(label12);
label13 = new java.awt.Label("Repair 5");
label13.setBounds(360,183,48,17);
label13.setEnabled(false);
panel1.add(label13);
label14 = new java.awt.Label("OBA Timers");
label14.setBounds(360,111,72,12);
label14.setEnabled(false);
panel1.add(label14);
textField2 = new java.awt.TextField();
textField2.setText("Start :");
textField2.setBounds(408,135,72,19);
textField2.setForeground(new Color(16776960));
textField2.setBackground(new Color(4210752));
panel1.add(textField2);
textField2.setEnabled(false);
textField3 = new java.awt.TextField();
textField3.setEditable(false);
textField3.setText("Start :");
textField3.setBounds(408,159,72,19);
textField3.setForeground(new Color(16776960));
textField3.setBackground(new Color(4210752));
panel1.add(textField3);
textField3.setEnabled(false);
textField4 = new java.awt.TextField();
textField4.setEditable(false);
textField4.setText("Start :");
textField4.setBounds(408,183,72,19);
textField4.setForeground(new Color(16776960));
textField4.setBackground(new Color(4210752));
panel1.add(textField4);
textField4.setEnabled(false);
bio_blood_chk = new java.awt.Checkbox("Blood");
bio_blood_chk.setBounds(492,135,67,18);
panel1.add(bio_blood_chk);
bio_blood_chk.setEnabled(false);
bio_blister_chk = new java.awt.Checkbox("Blister");
bio_blister_chk.setBounds(492,159,68,17);
panel1.add(bio_blister_chk);
bio_blister_chk.setEnabled(false);
```

```
bio_nerve_chk = new java.awt.Checkbox("Nerve");
bio_nerve_chk.setBounds(492,183,66,18);
panel1.add(bio_nerve_chk);
bio_nerve_chk.setEnabled(false);
label16 = new java.awt.Label("Bio/Chem Agent");
label16.setBounds(492,111,88,14);
label16.setEnabled(false);
panel1.add(label16);
panel2 = new java.awt.Panel();
panel2.setLayout(null);
panel2.setVisible(false);
panel2.setBounds(12,33,588,244);
panel2.setForeground(new Color(0));
panel2.setBackground(new Color(12632256));
tabPanel1.add(panel2);
rect1 = new symantec.itools.awt.shape.Rect();
rect1.setBounds(132,51,48,89);
panel2.add(rect1);
rect2 = new symantec.itools.awt.shape.Rect();
rect2.setBounds(192,51,48,89);
panel2.add(rect2);
rect3 = new symantec.itools.awt.shape.Rect();
rect3.setBounds(276,51,48,89);
panel2.add(rect3);
rect4 = new symantec.itools.awt.shape.Rect();
rect4.setBounds(336,51,48,89);
panel2.add(rect4);
rect5 = new symantec.itools.awt.shape.Rect();
try {
      rect5.setFillColor(new java.awt.Color(12632256));
}
catch(java.beans.PropertyVetoException e) { }
rect5.setBounds(156,15,72,24);
rect5.setForeground(new Color(0));
rect5.setBackground(new Color(12632256));
panel2.add(rect5);
rect6 = new symantec.itools.awt.shape.Rect();
rect6.setBounds(288,15,72,24);
panel2.add(rect6);
rect7 = new symantec.itools.awt.shape.Rect();
rect7.setBounds(420,15,72,24);
panel2.add(rect7);
rect8 = new symantec.itools.awt.shape.Rect();
rect8.setBounds(12,207,70,24);
panel2.add(rect8);
rect9 = new symantec.itools.awt.shape.Rect();
rect9.setBounds(96,207,71,24);
panel2.add(rect9);
rect10 = new symantec.itools.awt.shape.Rect();
rect10.setBounds(192,207,72,26);
panel2.add(rect10);
rect11 = new symantec.itools.awt.shape.Rect();
rect11.setBounds(288,207,72,25);
panel2.add(rect11);
rect12 = new symantec.itools.awt.shape.Rect();
rect12.setBounds(468,207,72,25);
panel2.add(rect12);
```

```
rect13 = new symantec.itools.awt.shape.Rect();
rect13.setBounds(384,207,72,25);
panel2.add(rect13);
rect14 = new symantec.itools.awt.shape.Rect();
rect14.setBounds(120,159,72,26);
panel2.add(rect14);
rect15 = new symantec.itools.awt.shape.Rect();
rect15.setBounds(228,159,72,24);
panel2.add(rect15);
rect16 = new symantec.itools.awt.shape.Rect();
rect16.setBounds(336,159,72,25);
panel2.add(rect16);
fwd_fuel_txt = new java.awt.TextField();
fwd_fuel_txt.setText("2-25-0-F");
fwd_fuel_txt.setBounds(468,75,96,20);
panel2.add(fwd_fuel_txt);
textField6 = new java.awt.TextField();
textField6.setText("SECURED");
textField6.setBounds(468,111,96,19);
panel2.add(textField6);
label15 = new java.awt.Label("Fwd Fuel");
label15.setBounds(396,75,60,16);
panel2.add(label15);
aft_Fuel_txt = new java.awt.Label("Aft Fuel");
aft_Fuel_txt.setBounds(396,111,49,16);
panel2.add(aft_Fuel_txt);
label17 = new java.awt.Label("NR 1 GTG");
label17.setBounds(168,3,60,12);
panel2.add(label17);
circle1 = new symantec.itools.awt.shape.Circle();
try {
      circle1.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle1.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle1.setBounds(180,15,24,24);
panel2.add(circle1);
circle2 = new symantec.itools.awt.shape.Circle();
try {
      circle2.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle2.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle2.setBounds(144,87,23,23);
panel2.add(circle2);
circle3 = new symantec.itools.awt.shape.Circle();
try {
      circle3.setFillColor(new java.awt.Color(255));
}
catch(java.beans.PropertyVetoException e) { }
try {
```

```
        circle3.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle3.setBounds(204,87,24,24);
panel2.add(circle3);
circle4 = new symantec.itools.awt.shape.Circle();
try {
        circle4.setFillColor(new java.awt.Color(16711680));
}
catch(java.beans.PropertyVetoException e) { }
try {
        circle4.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle4.setBounds(312,15,24,24);
panel2.add(circle4);
circle5 = new symantec.itools.awt.shape.Circle();
try {
        circle5.setFillColor(new java.awt.Color(16711680));
}
catch(java.beans.PropertyVetoException e) { }
try {
        circle5.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle5.setBounds(288,87,24,24);
panel2.add(circle5);
circle6 = new symantec.itools.awt.shape.Circle();
try {
        circle6.setFillColor(new java.awt.Color(16711680));
}
catch(java.beans.PropertyVetoException e) { }
try {
        circle6.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle6.setBounds(348,87,22,22);
panel2.add(circle6);
circle7 = new symantec.itools.awt.shape.Circle();
try {
        circle7.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
        circle7.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle7.setBounds(444,15,24,24);
panel2.add(circle7);
circle8 = new symantec.itools.awt.shape.Circle();
try {
        circle8.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
        circle8.setFillMode(true);
}
```

```
catch(java.beans.PropertyVetoException e) { }
circle8.setBounds(144,159,24,24);
panel2.add(circle8);
circle9 = new symantec.itools.awt.shape.Circle();
try {
      circle9.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle9.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle9.setBounds(252,159,24,24);
panel2.add(circle9);
circle10 = new symantec.itools.awt.shape.Circle();
try {
      circle10.setFillColor(new java.awt.Color(16711680));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle10.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle10.setBounds(360,159,24,24);
panel2.add(circle10);
circle11 = new symantec.itools.awt.shape.Circle();
try {
      circle11.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle11.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle11.setBounds(36,207,24,24);
panel2.add(circle11);
circle12 = new symantec.itools.awt.shape.Circle();
try {
      circle12.setFillColor(new java.awt.Color(255));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle12.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle12.setBounds(120,207,24,24);
panel2.add(circle12);
circle13 = new symantec.itools.awt.shape.Circle();
try {
      circle13.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle13.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle13.setBounds(216,207,24,24);
```

```
panel2.add(circle13);
circle14 = new symantec.itools.awt.shape.Circle();
try {
      circle14.setFillColor(new java.awt.Color(255));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle14.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle14.setBounds(312,207,24,24);
panel2.add(circle14);
circle15 = new symantec.itools.awt.shape.Circle();
try {
      circle15.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle15.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle15.setBounds(492,207,24,24);
panel2.add(circle15);
circle16 = new symantec.itools.awt.shape.Circle();
try {
      circle16.setFillColor(new java.awt.Color(16711680));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle16.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle16.setBounds(408,207,24,24);
panel2.add(circle16);
label18 = new java.awt.Label("Fire Pump 1");
label18.setBounds(12,195,72,9);
panel2.add(label18);
label19 = new java.awt.Label("Fire Pump 2");
label19.setBounds(96,195,67,9);
panel2.add(label19);
label20 = new java.awt.Label("Fire Pump 3");
label20.setBounds(192,195,72,11);
panel2.add(label20);
label21 = new java.awt.Label("Fire Pump 4");
label21.setBounds(288,195,72,12);
panel2.add(label21);
label22 = new java.awt.Label("Fire Pump 5");
label22.setBounds(384,195,74,10);
panel2.add(label22);
label23 = new java.awt.Label("Fire Pump 6");
label23.setBounds(468,195,72,12);
panel2.add(label23);
label24 = new java.awt.Label("SWS 1");
label24.setBounds(132,147,48,12);
panel2.add(label24);
label25 = new java.awt.Label("SWS 2");
label25.setBounds(240,147,45,12);
```

```
panel2.add(label25);
label26 = new java.awt.Label("SWS 3");
label26.setBounds(348,147,41,12);
panel2.add(label26);
label27 = new java.awt.Label("2B");
label27.setBounds(144,63,24,19);
panel2.add(label27);
label28 = new java.awt.Label("2A");
label28.setBounds(204,63,26,24);
panel2.add(label28);
label29 = new java.awt.Label("1B");
label29.setBounds(288,63,28,23);
panel2.add(label29);
label30 = new java.awt.Label("1A");
label30.setBounds(348,63,26,23);
panel2.add(label30);
label31 = new java.awt.Label("NR 2 GTG");
label31.setBounds(300,3,60,10);
panel2.add(label31);
label32 = new java.awt.Label("NR 3 GTG");
label32.setBounds(432,3,60,12);
panel2.add(label32);
circle17 = new symantec.itools.awt.shape.Circle();
try {
      circle17.setFillColor(new java.awt.Color(16711680));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle17.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle17.setBounds(84,15,24,24);
circle17.setForeground(new Color(0));
circle17.setBackground(new Color(16777215));
panel2.add(circle17);
circle18 = new symantec.itools.awt.shape.Circle();
try {
      circle18.setFillColor(new java.awt.Color(16776960));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle18.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle18.setBounds(84,51,24,24);
circle18.setForeground(new Color(0));
panel2.add(circle18);
circle19 = new symantec.itools.awt.shape.Circle();
try {
      circle19.setFillColor(new java.awt.Color(255));
}
catch(java.beans.PropertyVetoException e) { }
try {
      circle19.setFillMode(true);
}
catch(java.beans.PropertyVetoException e) { }
circle19.setBounds(84,87,24,24);
```

```
panel2.add(circle19);
label38 = new java.awt.Label("Damaged");
label38.setBounds(12,15,60,19);
panel2.add(label38);
label39 = new java.awt.Label("On-line");
label39.setBounds(12,51,65,20);
panel2.add(label39);
label40 = new java.awt.Label("Off-line");
label40.setBounds(12,87,68,21);
panel2.add(label40);
label41 = new java.awt.Label("Fuel tank on-line");
label41.setBounds(468,51,93,17);
panel2.add(label41);
panel3 = new java.awt.Panel();
panel3.setLayout(null);
panel3.setBounds(12,33,588,244);
tabPanel1.add(panel3);
list2 = new java.awt.List(4);
panel3.add(list2);
list2.addItemListener( this );
list2.setBounds(24,27,105,99);
list2.setBackground(new Color(16777215));
list3 = new java.awt.List(4);
list3.addItemListener( this );
panel3.add(list3);
list3.setBounds(144,27,105,99);
list3.setBackground(new Color(16777215));
list4 = new java.awt.List(4);
list4.addItemListener( this );
panel3.add(list4);
list4.setBounds(264,27,105,99);
list4.setBackground(new Color(16777215));
textField5 = new java.awt.TextField();
textField5.setBounds(24,183,108,20);
panel3.add(textField5);
textField7 = new java.awt.TextField();
textField7.setBounds(144,183,104,21);
panel3.add(textField7);
textField8 = new java.awt.TextField();
textField8.setBounds(264,183,109,22);
panel3.add(textField8);
label33 = new java.awt.Label("Repair 2 Team");
label33.setBounds(24,3,107,24);
panel3.add(label33);
label34 = new java.awt.Label("Repair 5 Team");
label34.setBounds(144,3,107,24);
panel3.add(label34);
label35 = new java.awt.Label("Repair 3 Team");
label35.setBounds(264,3,107,24);
panel3.add(label35);
rep2_addmem_btn = new java.awt.Button();
rep2_addmem_btn.setLabel("Add/Del Member");
rep2_addmem_btn.setBounds(24,135,107,20);
rep2_addmem_btn.setBackground(new Color(12632256));
rep2_addmem_btn.addActionListener( this );
panel3.add(rep2_addmem_btn);
rep5_addmem_btn = new java.awt.Button();
```

```
            rep5_addmem_btn.setLabel("Add/Del Member");
            rep5_addmem_btn.setBounds(144,135,107,20);
            rep5_addmem_btn.setBackground(new Color(12632256));
            rep5_addmem_btn.addActionListener( this );
            panel3.add(rep5_addmem_btn);
            rep3_addmem_btn = new java.awt.Button();
            rep3_addmem_btn.setLabel("Add/Del Member");
            rep3_addmem_btn.setBounds(264,135,107,20);
            rep3_addmem_btn.setBackground(new Color(12632256));
            rep3_addmem_btn.addActionListener( this );
            panel3.add(rep3_addmem_btn);
            label36 = new java.awt.Label("Enter Name and position");
            label36.setBounds(24,159,144,24);
            panel3.add(label36);
            label37 = new java.awt.Label("OBA Duration");
            label37.setBounds(24,207,84,23);
            panel3.add(label37);
            verticalLine1 = new
symantec.itools.awt.shape.VerticalLine();
            verticalLine1.setBounds(384,3,2,228);
            panel3.add(verticalLine1);
            label42 = new java.awt.Label("User Manager");
            label42.setBounds(444,3,84,24);
            panel3.add(label42);
            textField11 = new java.awt.TextField();
            textField11.setEchoChar('*');
            textField11.setBounds(468,63,114,19);
            panel3.add(textField11);
            textField10 = new java.awt.TextField();
            textField10.setBounds(468,27,114,19);
            panel3.add(textField10);
            textField12 = new java.awt.TextField();
            textField12.setEchoChar('*');
            textField12.setBounds(468,99,114,19);
            panel3.add(textField12);
            label43 = new java.awt.Label("User ID");
            label43.setBounds(408,27,48,19);
            panel3.add(label43);
            label44 = new java.awt.Label("Password");
            label44.setBounds(396,63,65,23);
            panel3.add(label44);
            label45 = new java.awt.Label("Pwd Verify");
            label45.setBounds(396,99,70,24);
            panel3.add(label45);
            user = new CheckboxGroup();
            radioButton1 = new java.awt.Checkbox("Priority User", user,
false);
            radioButton1.setBounds(480,135,96,18);
            panel3.add(radioButton1);
            radioButton2 = new java.awt.Checkbox("Config User", user,
false);
            radioButton2.setBounds(480,159,96,19);
            panel3.add(radioButton2);
            radioButton3 = new java.awt.Checkbox("View User", user,
false);
            radioButton3.setBounds(480,183,90,18);
            panel3.add(radioButton3);
```

```
        button2 = new java.awt.Button();
        button2.setLabel("New");
        button2.setBounds(396,135,72,24);
        button2.setBackground(new Color(12632256));
        panel3.add(button2);
        button3 = new java.awt.Button();
        button3.setLabel("View");
        button3.setBounds(396,171,72,24);
        button3.setBackground(new Color(12632256));
        panel3.add(button3);
        button4 = new java.awt.Button();
        button4.setLabel("Next ");
        button4.setBounds(492,207,72,24);
        button4.setBackground(new Color(12632256));
        panel3.add(button4);
        button5 = new java.awt.Button();
        button5.setLabel("Delete User");
        button5.setBounds(396,207,84,24);
        button5.setBackground(new Color(12632256));
        panel3.add(button5);
        textField9 = new java.awt.TextField();
        textField9.setText("45");
        textField9.setBounds(108,207,52,24);
        panel3.add(textField9);



        //}

}

//{{DECLARE_CONTROLS
JDBC01    theJDBC;        // The object that holds the results
TextField  theStatus      = new TextField(64);
String list2del [], list3del [], list4del [];
Object source1;


java.awt.Button new_Cas;
java.net.MulticastSocket S;
java.awt.Button fire_Btn;
java.awt.Button flood_Btn;
java.awt.Button nuclear_Btn;
java.awt.Button chem_btn;
java.awt.Button bio_Btn;
java.awt.Scrollbar horizontalScrollbar1;
symantec.itools.awt.TabPanel tabPanel1;
java.awt.Panel panel1;
java.awt.Checkbox x_Material_rb;
CheckboxGroup material_rbs;
java.awt.Checkbox z_Material_rb;
java.awt.Checkbox w_Material_rb;
java.awt.Label label1;
java.awt.Label label2;
java.awt.Checkbox class_A_rb;
CheckboxGroup fire_rbs;
```

```
java.awt.Checkbox class_B_rb;
java.awt.Checkbox class_C_rb;
java.awt.TextField textField1;
java.awt.Label label3;
java.awt.Label label4;
java.awt.Checkbox rep_2_chk;
java.awt.Checkbox rep_3_chk;
java.awt.Checkbox rep_5_chk;
java.awt.Label label5;
java.awt.Checkbox flood_maj_chk;
java.awt.Checkbox flood_min_chk;
java.awt.Checkbox flood_split_chk;
java.awt.Checkbox flood_pipe_chk;
java.awt.List list1;
java.awt.Label label6;
java.awt.Button halon_btn;
java.awt.TextField halon_time;
java.awt.Button button1;
java.awt.TextField bilge_start_time;
java.awt.TextField bilge_stop_time;
java.awt.Label label7;
java.awt.Label label8;
java.awt.Label label9;
java.awt.Label label10;
java.awt.Checkbox rep2_man_chk;
java.awt.Checkbox rep2_red_chk;
java.awt.Checkbox rep3_man_chk;
java.awt.Checkbox rep5_man_chk;
java.awt.Checkbox rep3_red_chk;
java.awt.Checkbox rep5_red_chk;
java.awt.Label label11;
java.awt.Label label12;
java.awt.Label label13;
java.awt.Label label14;
java.awt.TextField textField2;
java.awt.TextField textField3;
java.awt.TextField textField4;
java.awt.Checkbox bio_blood_chk;
java.awt.Checkbox bio_blister_chk;
java.awt.Checkbox bio_nerve_chk;
java.awt.Label label16;
java.awt.Panel panel2;
symantec.itools.awt.shape.Rect rect1;
symantec.itools.awt.shape.Rect rect2;
symantec.itools.awt.shape.Rect rect3;
symantec.itools.awt.shape.Rect rect4;
symantec.itools.awt.shape.Rect rect5;
symantec.itools.awt.shape.Rect rect6;
symantec.itools.awt.shape.Rect rect7;
symantec.itools.awt.shape.Rect rect8;
symantec.itools.awt.shape.Rect rect9;
symantec.itools.awt.shape.Rect rect10;
symantec.itools.awt.shape.Rect rect11;
symantec.itools.awt.shape.Rect rect12;
symantec.itools.awt.shape.Rect rect13;
symantec.itools.awt.shape.Rect rect14;
symantec.itools.awt.shape.Rect rect15;
```

```
symantec.itools.awt.shape.Rect rect16;
java.awt.TextField fwd_fuel_txt;
java.awt.TextField textField6;
java.awt.Label label15;
java.awt.Label aft_Fuel_txt;
java.awt.Label label17;
symantec.itools.awt.shape.Circle circle1;
symantec.itools.awt.shape.Circle circle2;
symantec.itools.awt.shape.Circle circle3;
symantec.itools.awt.shape.Circle circle4;
symantec.itools.awt.shape.Circle circle5;
symantec.itools.awt.shape.Circle circle6;
symantec.itools.awt.shape.Circle circle7;
symantec.itools.awt.shape.Circle circle8;
symantec.itools.awt.shape.Circle circle9;
symantec.itools.awt.shape.Circle circle10;
symantec.itools.awt.shape.Circle circle11;
symantec.itools.awt.shape.Circle circle12;
symantec.itools.awt.shape.Circle circle13;
symantec.itools.awt.shape.Circle circle14;
symantec.itools.awt.shape.Circle circle15;
symantec.itools.awt.shape.Circle circle16;
java.awt.Label label18;
java.awt.Label label19;
java.awt.Label label20;
java.awt.Label label21;
java.awt.Label label22;
java.awt.Label label23;
java.awt.Label label24;
java.awt.Label label25;
java.awt.Label label26;
java.awt.Label label27;
java.awt.Label label28;
java.awt.Label label29;
java.awt.Label label30;
java.awt.Label label31;
java.awt.Label label32;
symantec.itools.awt.shape.Circle circle17;
symantec.itools.awt.shape.Circle circle18;
symantec.itools.awt.shape.Circle circle19;
java.awt.Label label38;
java.awt.Label label39;
java.awt.Label label40;
java.awt.Label label41;
java.awt.Panel panel3;
java.awt.List list2;
java.awt.List list3;
java.awt.List list4;
java.awt.TextField textField5;
java.awt.TextField textField7;
java.awt.TextField textField8;
java.awt.Label label33;
java.awt.Label label34;
java.awt.Label label35;
java.awt.Button rep2_addmem_btn;
java.awt.Button rep5_addmem_btn;
java.awt.Button rep3_addmem_btn;
```

```
        java.awt.Label label36;
        java.awt.Label label37;
        symantec.itools.awt.shape.VerticalLine verticalLine1;
        java.awt.Label label42;
        java.awt.TextField textField11;
        java.awt.TextField textField10;
        java.awt.TextField textField12;
        java.awt.Label label43;
        java.awt.Label label44;
        java.awt.Label label45;
        java.awt.Checkbox radioButton1;
        CheckboxGroup user;
        java.awt.Checkbox radioButton2;
        java.awt.Checkbox radioButton3;
        java.awt.Button button2;
        java.awt.Button button3;
        java.awt.Button button4;
        java.awt.Button button5;
        java.awt.TextField textField9;
        //}
public void broadcast_update( )
    {
        try {
                //This is the hello socket
                byte[] msg = {'H', 'e', 'l', 'l', 'o'};

                InetAddress group = InetAddress.getByName("228.5.6.7");
                MulticastSocket s = new MulticastSocket(6789);
                s.joinGroup(group);
                DatagramPacket hi = new DatagramPacket(msg, msg.length,
                                    group, 6789);
                s.send(hi);
                 // get their responses!
                byte[] buf = new byte[1000];
                DatagramPacket recv = new DatagramPacket(buf,
buf.length);
                s.receive(recv);
                //...
                // OK, I'm done talking - leave the group...
                s.leaveGroup(group);

            }
    catch( java.io.IOException s) { }
////////////////////////////////////////////////////////////
}
public void start( )
    {
        theJDBC = new JDBC01(theStatus);
        try
        {
            theJDBC.openConnection(); //opens the socket/Bridge
        }
        catch (SQLException sql) { ; }

        try {
```

```
                theJDBC.executeQuery("SELECT * FROM rep_2_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list2.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }

      //Opens repair five member list
      try {

                theJDBC.executeQuery("SELECT * FROM rep_5_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list3.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }

      //Opens repair three member list

      try {

                theJDBC.executeQuery("SELECT * FROM rep_3_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list4.addItem(tokens.nextToken());

                  }
                  }
                  catch (SQLException sql) { ; }

    }

    public void stop( )
    {
        try
        {
            theJDBC.closeConnection( );
        }
        catch (SQLException sql) { ; }
    }

public void actionPerformed( ActionEvent e )
          {
          Object source  = e.getSource();


          if  (source == fire_Btn) {
```

```
            //Greys in the fire atributes

            x_Material_rb.setEnabled(true);
            z_Material_rb.setEnabled(true);
            w_Material_rb.setEnabled(true);
            rep2_red_chk.setEnabled(true);
            rep2_man_chk.setEnabled(true);
            rep3_red_chk.setEnabled(true);
            rep3_man_chk.setEnabled(true);
            rep5_red_chk.setEnabled(true);
            rep5_man_chk.setEnabled(true);
            textField1.setEnabled(true);
            list1.setEnabled(true);
            label1.setEnabled(true);
            label2.setEnabled(true);
            label3.setEnabled(true);
            label4.setEnabled(true);
            label6.setEnabled(true);
            label7.setEnabled(true);
            label8.setEnabled(true);
            label9.setEnabled(true);
            label10.setEnabled(true);
            label11.setEnabled(true);
            class_A_rb.setEnabled(true);
            class_B_rb.setEnabled(true);
            class_C_rb.setEnabled(true);
            label4.setEnabled(true);
            rep_2_chk.setEnabled(true);
            rep_3_chk.setEnabled(true);
            rep_5_chk.setEnabled(true);
            halon_btn.setEnabled(true);
            halon_time.setEnabled(true);
            button1.setEnabled(true);
            bilge_start_time.setEnabled(true);
            bilge_stop_time.setEnabled(true);
            label11.setEnabled(true);
          label12.setEnabled(true);
          label13.setEnabled(true);
          label14.setEnabled(true);
          textField2.setEnabled(true);
          textField3.setEnabled(true);
          textField4.setEnabled(true);
          try
           {
              //Still trying to get it to update
          theJDBC.executeQuery("SELECT elec_rep2 FROM dc_casualty
WHERE casualty_id = 1;");

            if (theJDBC.dumpResult()== "yes"){

                rep_2_chk.setState(true);
              }
            }
           catch (SQLException sql) { ; }

        }
```

```
if  (source == flood_Btn) {
   label5.setEnabled(true);
   flood_maj_chk.setEnabled(true);
   flood_min_chk.setEnabled(true);
   flood_split_chk.setEnabled(true);
   flood_pipe_chk.setEnabled(true);
   x_Material_rb.setEnabled(true);
    z_Material_rb.setEnabled(true);
    w_Material_rb.setEnabled(true);
    rep2_red_chk.setEnabled(true);
    rep2_man_chk.setEnabled(true);
    rep3_red_chk.setEnabled(true);
    rep3_man_chk.setEnabled(true);
    rep5_red_chk.setEnabled(true);
    rep5_man_chk.setEnabled(true);
    textField1.setEnabled(true);
    list1.setEnabled(true);
    label1.setEnabled(true);
    label7.setEnabled(true);
    label8.setEnabled(true);
    label9.setEnabled(true);
    label10.setEnabled(true);
}

if  (source == nuclear_Btn) {
    x_Material_rb.setEnabled(true);
    z_Material_rb.setEnabled(true);
    w_Material_rb.setEnabled(true);
    rep2_red_chk.setEnabled(true);
    rep2_man_chk.setEnabled(true);
    rep3_red_chk.setEnabled(true);
    rep3_man_chk.setEnabled(true);
    rep5_red_chk.setEnabled(true);
    rep5_man_chk.setEnabled(true);
    textField1.setEnabled(true);
    label1.setEnabled(true);
    label7.setEnabled(true);
    label8.setEnabled(true);
    label9.setEnabled(true);
    label10.setEnabled(true);

}
if  (source == chem_btn){
    x_Material_rb.setEnabled(true);
    z_Material_rb.setEnabled(true);
    w_Material_rb.setEnabled(true);
    rep2_red_chk.setEnabled(true);
    rep2_man_chk.setEnabled(true);
    rep3_red_chk.setEnabled(true);
    rep3_man_chk.setEnabled(true);
    rep5_red_chk.setEnabled(true);
    rep5_man_chk.setEnabled(true);
    textField1.setEnabled(true);
    label1.setEnabled(true);
    label7.setEnabled(true);
    label8.setEnabled(true);
```

```
        label9.setEnabled(true);
        label10.setEnabled(true);
        bio_blood_chk.setEnabled(true);
        bio_nerve_chk.setEnabled(true);
        bio_blister_chk.setEnabled(true);
        label16.setEnabled(true);
    }

    if  (source == bio_Btn) {
        x_Material_rb.setEnabled(true);
        z_Material_rb.setEnabled(true);
        w_Material_rb.setEnabled(true);
        rep2_red_chk.setEnabled(true);
        rep2_man_chk.setEnabled(true);
        rep3_red_chk.setEnabled(true);
        rep3_man_chk.setEnabled(true);
        rep5_red_chk.setEnabled(true);
        rep5_man_chk.setEnabled(true);
        textField1.setEnabled(true);
        label1.setEnabled(true);
        label7.setEnabled(true);
        label8.setEnabled(true);
        label9.setEnabled(true);
        label10.setEnabled(true);
        bio_blood_chk.setEnabled(true);
        bio_nerve_chk.setEnabled(true);
        bio_blister_chk.setEnabled(true);
        label16.setEnabled(true);


    }
    if  (source == new_Cas){


        material_rbs.setCurrent(null);
        fire_rbs.setCurrent(null);
        z_Material_rb.setState(false);
        w_Material_rb.setState(false);
        rep2_red_chk.setState(false);
        rep3_red_chk.setState(false);
        rep5_red_chk.setState(false);
        rep2_man_chk.setState(false);
        rep3_man_chk.setState(false);
        rep5_man_chk.setState(false);
        rep_2_chk.setState(false);
        rep_3_chk.setState(false);
        rep_5_chk.setState(false);
        class_A_rb.setState(false);
        class_B_rb.setState(false);
        class_C_rb.setState(false);
        bio_blood_chk.setState(false);
        bio_blister_chk.setState(false);
        bio_nerve_chk.setState(false);
        flood_maj_chk.setState(false);
        flood_min_chk.setState(false);
        flood_split_chk.setState(false);
```

```
    flood_pipe_chk.setState(false);
    textField1.setText("");
    list1.clear();
    bilge_start_time.setText("");
    bilge_stop_time.setText("");
    halon_time.setText("");

    rep2_man_chk.setEnabled(false);
    rep3_man_chk.setEnabled(false);
    rep5_man_chk.setEnabled(false);
    x_Material_rb.setEnabled(false);
    z_Material_rb.setEnabled(false);
    w_Material_rb.setEnabled(false);

    rep2_red_chk.setEnabled(false);
    rep2_man_chk.setEnabled(false);
    rep3_red_chk.setEnabled(false);
    rep3_man_chk.setEnabled(false);
    rep5_red_chk.setEnabled(false);
    rep5_man_chk.setEnabled(false);
    textField1.setEnabled(false);
    list1.setEnabled(false);
    label1.setEnabled(false);
    label2.setEnabled(false);
    label3.setEnabled(false);
    label4.setEnabled(false);
    label6.setEnabled(false);
    label7.setEnabled(false);
    label8.setEnabled(false);
    label9.setEnabled(false);
    label10.setEnabled(false);
    label11.setEnabled(false);
    class_A_rb.setEnabled(false);
    class_B_rb.setEnabled(false);
    class_C_rb.setEnabled(false);
    label4.setEnabled(false);
    rep_2_chk.setEnabled(false);
    rep_3_chk.setEnabled(false);
    rep_5_chk.setEnabled(false);
    halon_btn.setEnabled(false);
    halon_time.setEnabled(false);
    button1.setEnabled(false);
    bilge_start_time.setEnabled(false);
    bilge_stop_time.setEnabled(false);
    label11.setEnabled(false);
  label12.setEnabled(false);
  label13.setEnabled(false);
  label14.setEnabled(false);
  textField2.setEnabled(false);
  textField3.setEnabled(false);
  textField4.setEnabled(false);
    bio_blood_chk.setEnabled(false);
    bio_nerve_chk.setEnabled(false);
    bio_blister_chk.setEnabled(false);
    label16.setEnabled(false);
    label5.setEnabled(false);
    flood_maj_chk.setEnabled(false);
```

```
            flood_min_chk.setEnabled(false);
            flood_split_chk.setEnabled(false);
            flood_pipe_chk.setEnabled(false);

          }
         if  (source == halon_btn) {
            java.util.Date today = new java.util.Date();
            Timestamp  now = new Timestamp(today.getTime());
            String timer = now.toString();
            halon_time.setText(timer);

          }

         if ((source == button1) &
(bilge_start_time.getText().equals(""))){
                        java.util.Date today = new java.util.Date();
                        Timestamp  now = new
Timestamp(today.getTime());
                        String timer = now.toString();
                        bilge_start_time.setText(timer);
                    }
         else if ((source == button1) &
(!(bilge_start_time.getText().equals(""))){
                       java.util.Date today = new java.util.Date();
                        Timestamp  now = new
Timestamp(today.getTime());
                       String timer = now.toString();
                        bilge_stop_time.setText(timer);
                      }


          if  ((source == rep2_addmem_btn) && (source1 == list2)) {


              try
                 {
            theJDBC.executeUpdate("DELETE FROM rep_2_members WHERE
name_rate = "
                                      + "(" + "'" +
list2.getSelectedItem() + "'" + ")" + ";");


            //textField5.setText("");
                 }
              catch (SQLException sqlex) { }
            try {

             list2.clear();
             theJDBC.executeQuery("SELECT * FROM rep_2_members;");
             StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

             while (tokens.hasMoreTokens()){
               list2.addItem(tokens.nextToken());
```

```
                    }
                    }
                    catch (SQLException sql) { ; }




                }
            //Just the add

            if  (source == rep2_addmem_btn) {
                try
                    {

               theJDBC.executeUpdate("INSERT INTO rep_2_members VALUES "
                                     + "(" + "'" + textField5.getText()
+ "'" + ")" + ";");

            textField5.setText("");

                    }
                 catch (SQLException sqlex) { }
                try {
                list2.clear();
                theJDBC.executeQuery("SELECT * FROM rep_2_members;");
                StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                while (tokens.hasMoreTokens()){
                  list2.addItem(tokens.nextToken());
                    }
                    }
                  catch (SQLException sql) { ; }



               }


            if  ((source1 == list3) & (source == rep5_addmem_btn) ) {



                 try
                    {
               theJDBC.executeUpdate("DELETE FROM rep_5_members WHERE
name_rate = "
                                     + "(" + "'" +
list3.getSelectedItem() + "'" + ")" + ";");
```

```
                    }
                  catch (SQLException sqlex) { }
                try {

                 list3.clear();
                 theJDBC.executeQuery("SELECT * FROM rep_5_members;");
                 StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                 while (tokens.hasMoreTokens()){
                   list3.addItem(tokens.nextToken());

                   }
                   }
                   catch (SQLException sql) { ; }




            }


            if  (source == rep5_addmem_btn) {

                try
                    {

               theJDBC.executeUpdate("INSERT INTO rep_5_members VALUES "
                                     + "(" + "'" + textField7.getText()
+ "'" + ")" + ";");


               textField7.setText("");
                    }
                  catch (SQLException sqlex) { }
                 try {
                 list3.clear();
                 theJDBC.executeQuery("SELECT * FROM rep_5_members;");
                 StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

                 while (tokens.hasMoreTokens()){
                   list3.addItem(tokens.nextToken());
                     }
                     }
                   catch (SQLException sql) { ; }



               }

             if  ((source1 == list4) & (source == rep3_addmem_btn)) {
```

```
                try
                   {
               theJDBC.executeUpdate("DELETE FROM rep_3_members WHERE
name_rate = "
                                        + "(" + "'" +
list4.getSelectedItem() + "'" + ")" + ";");



                   }
                catch (SQLException sqlex) { }
              try {

               list4.clear();
               theJDBC.executeQuery("SELECT * FROM rep_3_members;");
               StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

               while (tokens.hasMoreTokens()){
                 list4.addItem(tokens.nextToken());

                 }
                 }
                 catch (SQLException sql) { ; }

          }

          if  (source == rep3_addmem_btn) {

              try
                   {

              theJDBC.executeUpdate("INSERT INTO rep_3_members VALUES "
                                        + "(" + "'" + textField8.getText()
+ "'" + ")" + ";");

              textField8.setText("");
                   }
                catch (SQLException sqlex) { }
               try {
               list4.clear();
               theJDBC.executeQuery("SELECT * FROM rep_3_members;");
               StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult(), "\n", false);

               while (tokens.hasMoreTokens()){
                 list4.addItem(tokens.nextToken());
                 }
                 }
                 catch (SQLException sql) { ; }



              }
```

```
        }
public void itemStateChanged (ItemEvent g)
  {
    source1 = g.getSource();
    list2del = list2.getSelectedItems();
    list3del = list3.getSelectedItems();
    list4del = list4.getSelectedItems();
  }
}
```

APPENDIX C. SOURCE CODE FOR MAINTENANCE APPLICATION

```
// Sorter View screen Applet
// CS3773 Java as a second Language Final Project
// Version 1.0
// Kurt Rothenhaus
// Applet allows the user to update and view maintenance actions
// for specified US Naval Vessels (Simulated)
// Contains no classified or sensitive data
// Programmer wishes to thank Java; how to program by
// Dietel & Dietel and
// SQL programming in Java by Bill McCarthy for their valuable
// code which I unabashadly used.
// Applet can be viewed at:
// http://web.nps.navy.mil/~kjrothen

import java.awt.*;
import java.applet.Applet;
import java.util.*;
import java.awt.event.*;
import j102.sql.*;
import Framekiloview;

public class Sorter extends Applet
             implements ActionListener, ItemListener
{


    JDBC01    theJDBC;        // The object that holds the results
    //GUI Declarations
    java.awt.List twokiloList;
      java.awt.Checkbox checkbox1;
      java.awt.Checkbox checkbox2;
      java.awt.Checkbox checkbox3;
      java.awt.Choice choice1;
      java.awt.Choice choice2;
      java.awt.TextField textField1;
      java.awt.Button button1;
      java.awt.Checkbox checkbox4;
      java.awt.TextField textField2;
      java.awt.Label label1;
      //Used for debugging displays SQL errors
      TextField  theStatus      = new TextField(64);
      JDBC02     theDB          = new JDBC02(theStatus);
    Framekiloview f;
    String currentquery [];

      public void init()
      {

            setLayout(null);
            setSize(526,366);
            setBackground(new Color(12632256));

            twokiloList = new java.awt.List(1000, false);
            twokiloList.setBounds(12,156,505,126);
            twokiloList.setBackground(new Color(12632256));
```

```
twokiloList.addActionListener( this );
twokiloList.addItemListener( this );
add(twokiloList);
checkbox1 = new java.awt.Checkbox("JSN Number");
checkbox1.setBounds(12,24,124,22);
checkbox1.setBackground(new Color(12632256));
add(checkbox1);
checkbox2 = new java.awt.Checkbox("Sort by WC");
checkbox2.setBounds(12,60,130,27);
checkbox2.setBackground(new Color(12632256));
add(checkbox2);
checkbox3 = new java.awt.Checkbox("Sort by UIC");
checkbox3.setBounds(12,96,127,20);
checkbox3.setBackground(new Color(12632256));
add(checkbox3);
choice1 = new java.awt.Choice();
choice1.addItem("CS01");
choice1.addItem("CS02");
choice1.addItem("CS03");
choice1.addItem("CS04");
choice1.addItem("EM01");
choice1.addItem("EM02");
choice1.addItem("FU01");
add(choice1);
choice1.setBounds(156,60,75,24);
choice1.setBackground(new Color(16777215));
choice2 = new java.awt.Choice();
choice2.addItem("29233");
choice2.addItem("39345");
choice2.addItem("33333");
add(choice2);
choice2.setBounds(156,96,75,24);
choice2.setBackground(new Color(16777215));
textField1 = new java.awt.TextField();
textField1.setBounds(384,48,126,24);
textField1.setBackground(new Color(12632256));
add(textField1);
button1 = new java.awt.Button();
button1.setLabel("Start Query");
button1.setBounds(312,96,120,32);
button1.setBackground(new Color(12632256));
button1.addActionListener( this );
add(button1);
checkbox4 = new java.awt.Checkbox("Seacrh for String");
checkbox4.setBounds(264,48,120,24);
checkbox4.setBackground(new Color(12632256));
add(checkbox4);
textField2 = new java.awt.TextField();
textField2.setBounds(156,24,71,21);
textField2.setBackground(new Color(12632256));
add(textField2);
label1 = new java.awt.Label("Sorting  Manager");
label1.setBounds(180,0,144,24);
label1.setFont(new Font("Dialog", Font.BOLD, 16));
label1.setBackground(new Color(12632256));
add(label1);
```

```
            f = new Framekiloview("Maintenance Resource Management
System Two Kilo View"
                                    , currentquery );

      }
public void start( )
    {
        theJDBC = new JDBC01(theStatus);
        try
        {
            theJDBC.openConnection(); //opens the socket/Bridge
        }
        catch (SQLException sql) { ; }
    }

    public void stop( )
    {
        try
        {
            theJDBC.closeConnection( );
        }
        catch (SQLException sql) { ; }
    }



public void actionPerformed(ActionEvent e)
    {
         if (checkbox1.getState())
            {

              try
                {

             theJDBC.executeQuery("SELECT UIC, WC, JSN, Noun_Name,
BLK_35 FROM temp WHERE JSN = "
                                        + textField2.getText() + ";");
             twokiloList.clear();
             StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult( ), "\n" , false);


             while ( tokens.hasMoreTokens() )
                twokiloList.addItem( tokens.nextToken());

                }
                catch (SQLException sql) { ; }
            }
         if (checkbox2.getState())
            {
                try
                {

                theJDBC.executeQuery("SELECT UIC, WC, JSN, Noun_Name,
BLK_35 FROM temp WHERE WC = "
                                            + "'" +
choice1.getSelectedItem().toString()+ "'" + ";");
```

```
                twokiloList.clear();
              StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult( ), "\n" , false);


              while ( tokens.hasMoreTokens() )
                 twokiloList.addItem( tokens.nextToken());

                 }
                 catch (SQLException sql) { ; }
              }
           if (checkbox3.getState())
              {
                 //search for an item by UIC
                 try
                 {

                 theJDBC.executeQuery("SELECT UIC, WC, JSN, Noun_Name,
BLK_35 FROM temp WHERE UIC = "
                                          + "'" +
choice2.getSelectedItem().toString()+ "'" + ";");
                 twokiloList.clear();
              StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult( ), "\n" , false);


              while ( tokens.hasMoreTokens() )
                 twokiloList.addItem( tokens.nextToken());

                 }
                 catch (SQLException sql) { ; }
              }
           if (checkbox4.getState())
              {
                 //search for a string in all the jobs

                try
                 {

              theJDBC.executeQuery("SELECT UIC, WC, JSN, Noun_Name,
BLK_35 FROM temp WHERE Noun_Name LIKE "
                                    + "'" + textField1.getText() + "%'" +
";");



              twokiloList.clear();
              StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult( ), "\n" , false);


              while ( tokens.hasMoreTokens() )
                 twokiloList.addItem( tokens.nextToken());

                 }
                 catch (SQLException sql) { ; }
```

```
            }
         if ( (checkbox4.getState()) & (checkbox2.getState()))
            {
                //search for a string in all the jobs and under a
certian WC
                try
                {

             theJDBC.executeQuery("SELECT UIC, WC, JSN, Noun_Name,
BLK_35 FROM temp WHERE Noun_Name LIKE "
                                    + "'" + textField1.getText() + "%'" +
" AND WC = "
                                        + "'" +
choice1.getSelectedItem().toString()+ "'" + ";");



            twokiloList.clear();
             StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult( ), "\n" , false);


             while ( tokens.hasMoreTokens() )
                twokiloList.addItem( tokens.nextToken());

                }
                catch (SQLException sql) { ; }
            }
         if (checkbox4.getState() & checkbox3.getState())
            {
                try
                {

             theJDBC.executeQuery("SELECT UIC, WC, JSN, Noun_Name,
BLK_35 FROM temp WHERE Noun_Name LIKE "
                                    + "'" + textField1.getText() + "%'" +
" AND UIC = "
                                        + "'" +
choice2.getSelectedItem().toString()+ "'" + ";");



             twokiloList.clear();
             StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult( ), "\n" , false);


             while ( tokens.hasMoreTokens() )
                twokiloList.addItem( tokens.nextToken());

                }
                catch (SQLException sql) { ; }
            }
         if (checkbox2.getState() & checkbox3.getState())
            {
                try
```

```
                {
             //Determines the jobs that have both WC UIC
             theJDBC.executeQuery("SELECT UIC, WC, JSN, Noun_Name,
BLK_35 FROM temp WHERE WC = "
                                    + "'" +
choice1.getSelectedItem().toString()+ "'" + " AND UIC = "
                                    + "'" +
choice2.getSelectedItem().toString()+ "'" + ";");

             twokiloList.clear();
             StringTokenizer tokens = new StringTokenizer
(theJDBC.dumpResult( ), "\n" , false);


             while ( tokens.hasMoreTokens() )
                twokiloList.addItem( tokens.nextToken());


            }
                catch (SQLException sql) { ; }
            }

       }

public void itemStateChanged ( ItemEvent e )
{
   // textField2.setText("testtesttest");
  f.setVisible ( true ); //show the
  currentquery = twokiloList.getSelectedItems();
}
}
// JDB01 CLass
// CS3773 Java as a second Language Final Project
// Version 1.0
// Kurt Rothenhaus
// CLASS automates a number of usefull functions that the sorter
// uses to display its results
// Contains no classified or sensitive data
// Programmer wishes to thank Java; how to program by
// Dietel & Dietel and
// SQL programming in Java by Bill McCarthy for their valuable
// code which I unabashadly used.
// Applet can be viewed at:
// http://web.nps.navy.mil/~kjrothen

import java.awt.*;
import java.util.Properties;
import java.net.URL;
import j102.sql.*;  //Creates abstract bridge for socket/JDBC
import java.applet.*;

public class JDBC02
{
    String            theSource   = "";
    String            theUser     = "";
    String            thePassword = "";
```

```
    Connection        theConnection  = null;
    DatabaseMetaData  theDBMetaData  = null;
    Statement         theStatement   = null;
    ResultSet         theResultSet   = null;
    ResultSetMetaData theMetaData    = null;
    int               theUpdateCount = -1;
    boolean           theResult;

    TextField theStatus;

    public JDBC02(TextField status)
    {
        theStatus = status;
    }

    public void openConnection()
    throws SQLException
    {

        try
        {
        IDSDriver drv = new j102.sql.IDSDriver(); // New ODBC driver
instansiated
            String url =
"jdbc:ids://131.120.27.56:12/conn?dsn='Maint2'";

            Connection theConnection = drv.connect(url,null); //connect
is really a java.sql


      //theConnection.setTransactionIsolation(Connection.TRANSACTION_REA
D_UNCOMMITTED);
            theDBMetaData
              = theConnection.getMetaData( );
            theStatement
              = theConnection.createStatement( );
            theResultSet = null;
            theMetaData  = null;
            theStatus.setText("Status: OK");
        }
        catch (SQLException sql)
        {
            handleError(sql);
        }
        //catch (ClassNotFoundException ex)
       // {
       //     handleError(ex);
       // }
    }

    public void closeConnection( )
    throws SQLException
    {
        try
        {
            if (theConnection != null)
                theConnection.close( );
```

```
        }
        catch (SQLException sql)  { handleError(sql); }
    }

    public boolean execute(String sql)
    throws SQLException
    {
        if (theResultSet != null)
            theResultSet.close( );
        theResultSet = null;
        theMetaData  = null;
        theUpdateCount = -1;
        theResult = theStatement.execute(sql);
        if (theResult)
        {
            theResultSet = theStatement.getResultSet( );
            if (theResultSet != null)
                theMetaData =  theResultSet.getMetaData( );
        }
        else
        {
            theUpdateCount = theStatement.getUpdateCount( );
        }
        return theResult;
    }

    public boolean getNextResult( )
    throws SQLException
    {
        theResultSet = null;
        theMetaData  = null;
        theUpdateCount = -1;
        theResult = theStatement.getMoreResults( );
        if (theResult)
        {
            theResultSet = theStatement.getResultSet( );
            if (theResultSet != null)
                theMetaData =  theResultSet.getMetaData( );
        }
        else
        {
            try
            {
                theUpdateCount = theStatement.getUpdateCount( );
            }
            catch (SQLException sql) { ; }
        }
        return theResult;
    }

    public void executeQuery(String sql)
    throws SQLException
    {
          if (theResultSet != null)
            theResultSet.close( );
          theResultSet = theStatement.executeQuery(sql);
          if (theResultSet != null)
```

```
        theMetaData  = theResultSet.getMetaData( );
}

public boolean nextRow( )
throws SQLException
{
    if (theResultSet != null)
        return theResultSet.next( );
    return false;
}

public void closeResultSet( )
throws SQLException
{
    try
    {
        if (theResultSet != null) theResultSet.close( );
    }
    catch (SQLException sql) { handleError(sql); }
}

public int executeUpdate(String sql)
throws SQLException
{
    int result = -1;
    try
    {
        if (theResultSet != null)
            theResultSet.close( );
        theResultSet = null;
        theMetaData = null;
        result = theStatement.executeUpdate(sql);
    }
    catch (SQLException e) { handleError(e); }
    return result;
}

public String dumpResult( )
throws SQLException
{
    if (theResultSet == null) return "";
    String result = "";
    try
    {
        int column_count = theMetaData.getColumnCount( );
        while (theResultSet.next( ))
        {
            boolean first = true;
            for (int i = 1; i <= column_count; i++)
            {
                if (!first) result += ", ";
                result += theResultSet.getString(i);
                first = false;
            }
            result += "\n";
        }
    }
```

```
    catch (SQLException sql) { handleError(sql); }
    return result;
}

String getFieldList(String [ ] fields)
{
    String result = "(";
    boolean first = true;
    for (int i = 0; i < fields.length; i++)
    {
        if (!first) result += ", ";
        first = false;
        result += fields[i];
    }
    result += ") ";
    return result;
}

String getValueList(String [ ] values, boolean [ ] isQuoted)
{
    String result = "VALUES (";
    boolean first = true;
    for (int i = 0; i < values.length; i++)
    {
        if (!first) result += ", ";
        first = false;
        String value = values[i];
        if (isQuoted[i])
        {
            result += "'";

            // double any embedded single quotes
            int j;
            while ((j = value.indexOf('\'')) >= 0)
            {
                if (j > 0)
                {
                    result += value.substring(0, j);
                }
                result += "''";
                if (value.length( ) > j + 1)
                {
                    value = value.substring(j + 1);
                }
                else
                {
                    value = "";
                }
            }
            result += value + "'";
        }
        else
        {
            result += value;
        }
    }
    result += ") ";
```

```
        return result;
    }

    String getNonNullString(int col)
    throws SQLException
    {
        return nonNull(theResultSet.getString(col));
    }

    String nonNull(String s)
    {
        if (s != null) return s;
        return "";
    }

    public void handleError(Throwable t)
    throws SQLException
    {
        theStatus.setText("Error: " + t.getMessage( ));
        t.printStackTrace( );
        throw new SQLException(t.getMessage( ));
    }
}
// Frame view for TWO-KILO View screen Applet
// CS3773 Java as a second Language Final Project
// Version 1.0
// Kurt Rothenhaus
// Applet allows the user to update and view maintenance actions
// for specified US Naval Vessels (Simulated)
// Contains no classified or sensitive data
// Programmer wishes to thank Java; how to program by
// Dietel & Dietel and
// SQL programming in Java by Bill McCarthy for their valuable
// code which I unabashadly used.
// Applet can be viewed at:
// http://web.nps.navy.mil/~kjrothen


import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import j102.sql.*;
import java.util.*;
import java.util.Properties;
import java.net.URL;

public class Framekiloview extends Frame
                  implements ActionListener
{
    static final int BROWSING_FORWARD  = +1;
    static final int BROWSING_BACKWARD = -1;
    static final int FIELD_COUNT       =  8;

    JDBC01    theJDBC;
    TextField  UIC           = new TextField("",  5);
    TextField  WC            = new TextField("",  4);
```

```
TextField  JSN           = new TextField("",  6);
TextField  URG           = new TextField("",  1);
TextField  RDD           = new TextField("", 15);
TextField  Location      = new TextField("", 10);
TextField  Issue         = new TextField("", 15);
TextField  Noun_Name     = new TextField("", 25);
TextField  APL           = new TextField("", 10);
TextField  Qty           = new TextField("",  5);
TextField  NIS           = new TextField("",  1);
TextField  Blk_35        = new TextField("", 100);
TextField  FRate         = new TextField("",  5);
TextField  FF_Name       = new TextField("", 18);
TextField  FL_Name       = new TextField("", 18);


TextField [ ] theFields  =
  { UIC, WC, JSN, URG, RDD, Location, Issue, Noun_Name,
    APL, Qty, NIS, Blk_35, FRate, FF_Name, FL_Name};


Button     theFirstButton = new Button("<<");
Button     thePrevButton  = new Button("<");
Button     theNextButton  = new Button(">");
Button     theLastButton  = new Button(">>");
Button     theNewButton   = new Button("New");
Button     theSaveButton  = new Button("Save");
Button     theQuitButton  = new Button("Exit");

java.awt.Label label1;
java.awt.Label label2;
java.awt.Label label3;
java.awt.Label label4;
  java.awt.Label label5;
java.awt.Label label6;
  java.awt.Label label7;
java.awt.Label label8;
  java.awt.Label label9;
  java.awt.Label label10;
  java.awt.Label label11;
  java.awt.Label label12;
  java.awt.Label label13;
  java.awt.Label label14;
java.awt.Label label15;
java.awt.Label label20;
java.awt.Label label21;
java.awt.Label label22;


//Panel      theStatusPanel = new Panel(new BorderLayout( ));
TextField  theStatus      = new TextField(64);

JDBC02     theDB          = new JDBC02(theStatus);

DBRecord   theDBRecord    = new DBRecord(FIELD_COUNT);

int        theBrowseDirection = BROWSING_FORWARD;
```

```
boolean    haveResultSet  = false;
String currentjob [];

public Framekiloview( String s , String currentjob [])
{
    super( s );
    setSize( 650, 500 );
    addWindowListener( new CloseWindow() );
    setBackground(new Color(12632256));
    addWindowListener( new CloseWindow() );
    UIC.setBounds(96,36,72,24);
        add(UIC);
    label1 = new java.awt.Label("1. UIC");
        label1.setBounds(12,36,72,27);
        add(label1);
        label2 = new java.awt.Label("2. WC");
        label2.setBounds(180,36,36,27);
        add(label2);
        WC.setBounds(228,36,60,23);
        add(WC);
        label3 = new java.awt.Label("3. JSN");
        label3.setBounds(300,36,36,27);
        add(label3);
        JSN.setBounds(348,36,48,24);
        add(JSN);
        label4 = new java.awt.Label("4. URG");
        label4.setBounds(420,36,48,27);
        add(label4);
        URG.setBounds(492,36,36,23);
        add(URG);
        label5 = new java.awt.Label("5. RDD");
        label5.setBounds(12,72,48,27);
        add(label5);
        RDD.setBounds(60,72,72,24);
        add(RDD);
        Location.setBounds(216,72,84,24);
        add(Location);
        label6 = new java.awt.Label("6. Location");
        label6.setBounds(144,72,72,27);
        add(label6);
        label7 = new java.awt.Label("8. Issue Date");
        label7.setBounds(396,72,84,24);
        add(label7);
        Issue.setBounds(492,72,96,24);
        add(Issue);
        Noun_Name.setBounds(108,108,168,24);
        add(Noun_Name);
        label8 = new java.awt.Label("9. Noun Name");
        label8.setBounds(12,108,84,27);
        add(label8);
        APL.setBounds(468,108,168,24);
        add(APL);
        label9 = new java.awt.Label("11. APL");
        label9.setBounds(408,108,48,24);
        add(label9);
        label10 = new java.awt.Label("12. QTY");
        label10.setBounds(12,144,84,24);
```

```
        add(label10);
        Qty.setBounds(108,144,72,24);
        add(Qty);
        label15 = new java.awt.Label("NIS:");
        label15.setBounds(200,144,84,24);
        add(label15);
        NIS.setBounds(280,144,72,24);
        add(NIS);
        Blk_35.setBounds(24,204,612,72);
        add(Blk_35);
        label12 = new java.awt.Label("Blk. 35");
        label12.setBounds(12,180,84,24);
        add(label12);

        FRate.setBounds(72,288,66,24);
        add(FRate);
        label20 = new java.awt.Label("Rate:");
        label20.setBounds(36,288,36,24);
        add(label20);
        label21 = new java.awt.Label("F_Name");
        label21.setBounds(144,288,60,24);
        add(label21);
        FF_Name.setBounds(204,288,72,24);
        add(FF_Name);
        label22 = new java.awt.Label("L_Name");
        label22.setBounds(288,288,60,24);
        add(label22);
        FL_Name.setBounds(348,288,122,24);
        add(FL_Name);
        label13 = new java.awt.Label("Two-Kilo View Screen");
        label13.setBounds(240,6,204,19);
        label13.setFont(new Font("Dialog", Font.BOLD, 18));
        add(label13);


    // Buttons
    theFirstButton.setBounds(48,324,48,25);
    add(theFirstButton);
    theNextButton.setBounds(168,324,48,25);
    add(theNextButton);
        thePrevButton.setBounds(108,324,48,25);
        add(thePrevButton);
        theLastButton.setBounds(228,324,48,25);
        add(theLastButton);
        theNewButton.setBounds(288,324,48,25);
        add(theNewButton);
        theSaveButton.setBounds(348,324,48,25);
        add(theSaveButton);
        theQuitButton.setBounds(408,324,72,25);
        add(theQuitButton);


    theFirstButton.addActionListener(this);
    thePrevButton .addActionListener(this);
    theNextButton .addActionListener(this);
    theLastButton .addActionListener(this);
    theNewButton  .addActionListener(this);
```

```
        theSaveButton .addActionListener(this);
        theQuitButton .addActionListener(this);


        try
        {


            theDB.openConnection();
            theDBRecord = getFirstRecord( );

            //getFirstRecord( );
            if (theDBRecord != null)
            {
                theDBRecord.moveToScreen(theFields);


            }
        }
        catch (Exception e)
        {
            handleException(e);
        }

}

    public void stop( )
    {
        try
        {
            theDB.closeConnection( );
        }
        catch (Exception e)
        {
            handleException(e);
        }
        setVisible(false);
    }

    public void actionPerformed(ActionEvent event)
    {
        statusOK( );
        Object source = event.getSource( );

        if (source == theFirstButton)
        {
            DBRecord first = getFirstRecord( );
            if (first != null)
            {
                theDBRecord = first;
                theDBRecord.moveToScreen(theFields);

            }
            else
                noRecordFound( );
        }
        else if (source == thePrevButton)
```

```
    {
        DBRecord prev = getPrevRecord( );
        if (prev != null)
        {
            theDBRecord = prev;
            theDBRecord.moveToScreen(theFields);

        }
        else
            noRecordFound( );
    }
    else if (source == theNextButton)
    {
        DBRecord next = getNextRecord( );
        if (next != null)
        {
            theDBRecord = next;
            theDBRecord.moveToScreen(theFields);

        }
        else
            noRecordFound( );
    }
    else if (source == theLastButton)
    {
        DBRecord last = getLastRecord( );
        if (last != null)
        {
            theDBRecord = last;
            theDBRecord.moveToScreen(theFields);

        }
        else
            theDBRecord.moveToScreen(theFields);
    }
    else if (source == theNewButton)
    {
        newRecord( );
    }
    else if (source == theSaveButton)
    {
        saveRecord( );
    }

    else if (source == theQuitButton)
    {
        stop( );
        //CloseWindow();
    }
}

public DBRecord getFirstRecord( )
{
    DBRecord result = null;

    theBrowseDirection = BROWSING_FORWARD;
```

```
        try
        {

            theDB.executeQuery("SELECT * FROM temp ORDER BY JSN;");

            if (theDB.nextRow( ))
            {
                result = new DBRecord(theDB.theResultSet);
                haveResultSet = true;
            }
            else
                noRecordFound( );
        }
        catch (Exception e)
        {
            handleException(e);
        }

        return result;
    }

    public DBRecord getLastRecord( )
    {
        DBRecord result = null;

        theBrowseDirection = BROWSING_FORWARD;
        haveResultSet = false;

        try
        {
            theDB.executeQuery("SELECT * FROM temp "
              + " ORDER BY JSN;");

            if (theDB.nextRow( ))
            {
                do
                {
                    result = new DBRecord(theDB.theResultSet);
                }
                while (theDB.nextRow( ));
            }
            else noRecordFound( );
        }
        catch (Exception e)
        {
            handleException(e);
        }

        return result;
    }

    public DBRecord getPrevRecord( )
    {
        DBRecord result = null;

        try
        {
```

```
            if (!haveResultSet || theBrowseDirection !=
BROWSING_BACKWARD)
            {
                theBrowseDirection = BROWSING_BACKWARD;

                theDB.executeQuery("SELECT * FROM temp "
                  + "WHERE JSN < "
                  + getJSN( )
                  + " ORDER BY JSN DESC;");
            }

            if (theDB.nextRow( ))
            {
                result = new DBRecord(theDB.theResultSet);
                haveResultSet = true;
            }
            else
                noRecordFound( );
        }
        catch (Exception e)
        {
            handleException(e);
        }

        return result;
    }

    public DBRecord getNextRecord( )
    {
        DBRecord result = null;

        try
        {
            if (!haveResultSet || theBrowseDirection !=
BROWSING_FORWARD)
            {
                theBrowseDirection = BROWSING_FORWARD;

                theDB.executeQuery("SELECT * FROM temp "
                  + "WHERE temp > "
                  + getJSN( )
                  + " ORDER BY JSN;");
            }
            if (theDB.nextRow( ))
            {
                result = new DBRecord(theDB.theResultSet);
                haveResultSet = true;
            }
            else
                noRecordFound( );
        }
        catch (Exception e)
        {
            handleException(e);
        }

        return result;
```

```
    }


    public void newRecord( )
    {
        clearScreen( );
        try
        {
            theDB.closeResultSet( );
        }
        catch (SQLException sql) { ; }
        haveResultSet = false;
    }

    public void saveRecord( )
    {
        haveResultSet = false;
        String sql;
        if (JSN.getText( ).length( ) > 0)
        {
            sql = "DELETE FROM temp WHERE JSN = ";
            sql += JSN.getText( );
            sql += ";";
            try
            {
                theDB.executeUpdate(sql);
            }
            catch (SQLException e) { ; }

            sql = "INSERT INTO temp ";
            String [ ] fields = new String[theFields.length];
            for (int i = 0; i < fields.length; i++)
                fields [i] = theFields [i].getText( );

            boolean [] quotes = new boolean [fields.length];
            for (int i = 0; i < fields.length; i++)
                quotes  [i] = true;

            sql += theDB.getValueList(fields, quotes);
        }
        else
        {
            sql = "INSERT INTO temp ";
            sql += "(UIC, WC, JSN, URG, RDD, Location, Issue, Noun_Name,
APL, Qty, NIS, Blk_35, FRate, FF_Name, FL_Name) ";

            String [ ] fields = new String[theFields.length - 1];
            for (int i = 0; i < fields.length; i++)
                fields [i] = theFields [i + 1].getText( );

            boolean [] quotes = new boolean [fields.length];
            for (int i = 0; i < fields.length; i++)
                quotes  [i] = true;

            sql += theDB.getValueList(fields, quotes);
        }
```

```
        sql += ";";
        try
        {
            theDB.executeUpdate(sql);
            clearScreen( );

            DBRecord next = null;
            if (theBrowseDirection == BROWSING_FORWARD)
                next = getNextRecord( );
            else
                next = getPrevRecord( );
            if (next != null)
            {
                theDBRecord = next;
                theDBRecord.moveToScreen(theFields);
            }
            else
                noRecordFound( );
        }
        catch (SQLException sqlex)
        {
             handleException(sqlex);
        }
    }

    public String getJSN( )
    {
        if (theDBRecord == null) return "0";
        if (theDBRecord.theFields[2] == null) return "0";
        return theDBRecord.theFields[2];
    }
    public String getNIS( )
    {

        if (theDBRecord == null) return "0";
        if (theDBRecord.theFields[10] == null) return "0";
        return theDBRecord.theFields[10];


    }
    public void clearScreen( )
    {
        for (int i = 0; i < theFields.length; i++)
            theFields[i].setText("");
    }

    public void noRecordFound( )
    {
        setStatus("Status: No record found.");
    }

    public void statusOK( )
    {
        setStatus("Status: OK");
    }

    public void setStatus(String s)
```

```
    {
        theStatus.setText(s);
    }

    public void handleException(Exception e)
    {
        e.printStackTrace( );
    }


}


class DBRecord
{
    String [] theFields;

    public DBRecord(int fields)
    {
        theFields = new String [fields];
    }

    public DBRecord(ResultSet rs)
    throws SQLException
    {
        ResultSetMetaData meta = rs.getMetaData( );
        int fields = meta.getColumnCount( );
        theFields = new String[fields];

        for (int i = 1; i <= fields; i++)
        {
            theFields [i - 1] = nonNull(rs.getString(i));
        }
    }

    public DBRecord(TextField [ ] txt)
    {
        theFields = new String[txt.length];

        for (int i = 1; i <= theFields.length; i++)
        {
            theFields [i - 1] = (txt [i - 1]).getText( );
        }
    }

    public void moveFromResultSet(ResultSet rs, TextField [ ] txt)
    throws SQLException
    {
        loadFromResultSet(rs);
        moveToScreen(txt);
    }

    public void loadFromResultSet(ResultSet rs)
    throws SQLException
    {
        for (int i = 1; i <= theFields.length; i++)
        {
```

```
            theFields [i - 1] = nonNull(rs.getString(i));
        }
    }

    public void moveFromScreen(TextField [ ] txt)
    {
        for (int i = 1; i <= theFields.length; i++)
        {
            theFields [i - 1] = (txt [i - 1]).getText( );
        }
    }

    public void moveToScreen(TextField [ ] txt)
    {
        for (int i = 1; i <= theFields.length; i++)
        {
            (txt [i - 1]).setText(theFields [i - 1]);
        }
    }

    public String nonNull(String s)
    {
        if (s != null) return s;
        return "";
    }
}
```

LIST OF REFERENCES

1. Geier, J., *Wireless Networking Handbook*, New Riders Publishing, Indianapolis, ID, 1996.

2. Moss, K., *Java Servlets*, McGraw-Hill, New York, NY, 1998.

3. Zukowski, J., *Mastering Java 1.2*l, Sybex, San Francisco, CA, 1998.

4. McCarthy, B., *SQL Database Programming with Java*, The Coriolis Group, Scottsdale, AZ, 1998.

5. Ames, A., Nadeau, D., Moreland, J., *VRML Sourcebook*, John Wiley & Sons, New York, NY, 1997.

6. Tanenbaum, A., *Computer Networks 3rd ed*, Prentice Hall, Upper Saddle River, NJ, 1994.

7. Berzins, V., Luqi, L., *Software Engineering with Abstractions*, Addison-Wesley Pub Co, Reading, MA, 1990.

8. Walker, G., "Pen Computers in HealthCare," *Pen Computing*, Feb. 1999

9. Lawton, G., "Vendors Battle Over Mobile OS Market," *IEEE Computer Magazine*, Feb. 1999.

10. Chroust, L., " Overview of the IEEE 802.11 WLAN Standard," *news.medicaldesignonline.com/feature-articles/19980420-184.html.*

11. Lucent Technologies, "WaveLAN & IEEE 802.11," *ftp.wavelan.com/pub/pdf_file/ieee/sb_ieee.pdf*, 1998

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code CS 1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Professor Xiaoping Yun, Code EC/YX 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5118

5. Professor Ted Lewis, Code CS/TL 1
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Mr. Steve Lose 2
   Program Executive Officer, Submarines
   PMS 450T2, NC2 5W64
   2531 Jefferson Davis Highway
   Arlington, VA 22242-5168

7. LT Kurt Rothenhaus 2
   531 Main St. apt 911
   New York, NY 10044

8. Mr. Gary Lacombe 2
   171 Branch Hill Rd
   Preston, CT 06365